BUKU PINTAR INTERNET

TCP/IP

Standar, Desain, dan Implementasi

ONNO W. PURBO
(Pakar Internet)

Adnan Basalamah, Ismail Fahmi, dan
Achmad Husni Thamrin

Buku Pintar Internet

TCP/IP

Buku Pintar Internet

TCP/IP

Onno W. Purbo (Pakar Internet)
Adnan Basalamah
Ismail Fahmi
Achmad Husni Thamrin

Kerja sama
Penerbit PT Elex Media Komputindo
Kelompok Gramedia - Jakarta dengan
Computer Network Research Group
Institut Teknologi Bandung

Buku Pintar Internet
TCP/IP
Onno W. Purbo, Adnan Basalamah, Ismail Fahmi dan
Achmad Husni Thamrin
12198190
ISBN - 979-20-0759-8

Diterbitkan pertama kali oleh
Penerbit PT Elex Media Komputindo
Kelompok Gramedia, Anggota IKAPI, Jakarta 1998

Dicetak oleh Percetakan PT Gramedia, Jakarta
isi di luar tanggung jawab percetakan

Kata Pengantar

Teknologi jaringan komputer dan internet saat ini telah merasuk hampir ke seluruh segi kehidupan kehidupan. Sangat sulit pada saat ini menemukan bidang yang belum tersentuh oleh teknologi jaringan komputer. Proses desain dan administrasi jaringan komputer ini tentunya memerlukan pengetahuan khusus tentang teknologi internet. Dengan semakin maraknya jaringan komputer dan internet ini, kebutuhan akan ilmu dasarnya pun akan semakin meningkat.

Walaupun demikian, belum ada buku khusus dalam bahasa indonesia, yang membahas bagaimana internet bekerja sebagai suatu sistem yang lengkap. Kebanyakan buku hanya membahas Internet secara global, atau membahas secara detail namun hanya pada sisi client dan server saja. Jarang, atau bahkan tidak ada satu buku pun dalam bahasa indonesia yang membahas pengalokasian IP Address secara efisien, *Variabel Length Subnet Masking* (VLSM), atau teknik teknik dasar routing di internet. Kehadiran buku ini di tengah tengah pembaca sekalian diharapkan dapat mengisi kekosongan tersebut.

Buku ini sendiri merupakan kompilasi dari beberapa dokumen RFC (*Request For Comment*) yang kami anggap dapat memberikan pengetahuan mendasar mengenai internet: arsitektur TCP/IP, IP dan pengalamatan host, DNS, routing, beberapa protokol aplikasi, dan SNMP. kompilasi atas RFC ini dilakukan tak lain karena standar-standar protokol internet ditulis di dokumen RFC.

Kami juga berusaha membumi dengan memberi pendekatan praktis melalui beberapa contoh implementasi di sistem Unix yang berjalan di atas platform PC x86. Dengan demikian, pembaca dapat mencoba sendiri berdasarkan contoh konfigurasi yang kami berikan.

Dengan selesainya satu buku ini, yang InsyaAllah akan disusul pula dengan buku buku lainnya, kami mengucapkan terimakasih kepada seluruh pihak yang selama ini telah memberikan bantuan baik secara langsung maupun tidak langsung , dan juga bekerja sama dengan lembaga riset kami, Computer Network Research Group ITB. Segala bantuan tersebut telah memungkinkan kami untuk akhirnya dapat mengumpulkan hampir seluruh ilmu dasar jaringan komputer TCP/IP dalam buku ini.

Sebagai penutup, kami menyadari bahwa apa yang tercantum dalam buku ini belumlah sempurna. Jika anda merasakan ada hal yang kurang jelas dalam buku ini, atau anda ingin mengetahui lebih lanjut tentang jaringan komputer, anda dapat bertanya dengan mengirim email ke alamat kami berikut ini:

cnrg@itb.ac.id

Daftar Isi

Bab 1
Pendahuluan

1.1. TCP/IP dan Internet

Dalam dunia komunikasi data komputer, protokol mengatur bagaimana sebuah komputer berkomunikasi dengan komputer lain. Dalam jaringan komputer kita dapat menggunakan banyak macam protokol tetapi agar dua buah komputer dapat berkomunikasi, keduanya perlu menggunakan protokol yang sama. Protokol berfungsi mirip dengan bahasa. Agar dapat berkomunikasi, orang-orang perlu berbicara dan mengerti bahasa yang sama.

TCP/IP (*Transmission Control Protocol/Internet Protocol*) adalah sekelompok protokol yang mengatur komunikasi data komputer di Internet. Komputer-komputer yang terhubung ke Internet berkomunikasi dengan protokol ini. Karena menggunakan bahasa yang sama, yaitu protokol TCP/IP, perbedaan jenis komputer dan sistem operasi tidak menjadi masalah. Komputer PC dengan sistem operasi Windows dapat berkomunikasi dengan komputer Macintosh atau dengan Sun SPARC yang menjalankan Solaris. Jadi, jika sebuah komputer menggunakan protokol TCP/IP dan terhubung langsung ke Internet, maka komputer tersebut

dapat berhubungan dengan komputer di belahan dunia mana pun yang juga terhubung ke Internet.

Perkembangan TCP/IP yang diterima luas dan praktis menjadi standar de-facto jaringan komputer berkaitan dengan ciri-ciri yang terdapat pada protokol itu sendiri:

- Protokol TCP/IP dikembangkan menggunakan standar protokol yang terbuka.
- Standar protokol TCP/IP dalam bentuk *Request For Comment* (RFC) dapat diambil oleh siapapun tanpa biaya.
- TCP/IP dikembangkan dengan tidak tergantung pada sistem operasi atau perangkat keras tertentu.
- Pengembangan TCP/IP dilakukan dengan konsensus dan tidak tergantung pada *vendor* tertentu.
- TCP/IP independen terhadap perangkat keras jaringan dan dapat dijalankan pada jaringan Ethernet, Token Ring, jalur telepon dial-up, jaringan X.25, dan praktis jenis media transmisi apa pun.
- Pengalamatan TCP/IP bersifat unik dalam skala global. Dengan cara ini, komputer dapat saling terhubung walaupun jaringannya seluas Internet sekarang ini.
- TCP/IP memiliki fasilitas routing yang memungkinkan sehingga dapat diterapkan pada internetwork.
- TCP/IP memiliki banyak jenis layanan.

1.2. Standar TCP/IP dan Proses RFC

Internet dapat terbentuk karena sekumpulan besar jaringan komputer memiliki kesepakatan untuk *berbicara dalam bahasa yang sama*. Kesepakatan ini semata-mata merupakan kesepakatan yang bersifat teknis. Karenanya tidak ada suatu badan pun di dunia ini yang berhak mengatur jalannya Internet secara keseluruhan. Yang dapat diatur dalam Internet ialah protokol yang digunakan.

1.2.1. Badan Pengatur Internet

Ada empat badan yang bertanggung jawab dalam mengatur, mengontrol serta melakukan standarisasi protokol yang digunakan di Internet:

- *Internet Society* (ISOC). ISOC merupakan badan profesional yang memfasilitasi, mendukung, serta mempromosikan pertumbuhan Internet, sebagai infrastruktur komunikasi global untuk riset/ penelitian. Badan ini berurusan dengan tidak hanya dengan aspek-aspek teknis, namun juga aspek politik dan sosial dari jaringan Internet.
- *Internet Architecture Board* (IAB). Ialah badan koordinasi dan penasehat teknis bagi *Internet Society*. Badan ini betindak sebagai badan review teknis dan editorial akhir semua standar Internet. IAB memiliki otoritas untuk menerbitkan dokumen standar Internet yang dikenal sebagai RFC (*Request for Comment*). Tugas lain dari IAB ialah mengatur angka-angka dan konstanta yang

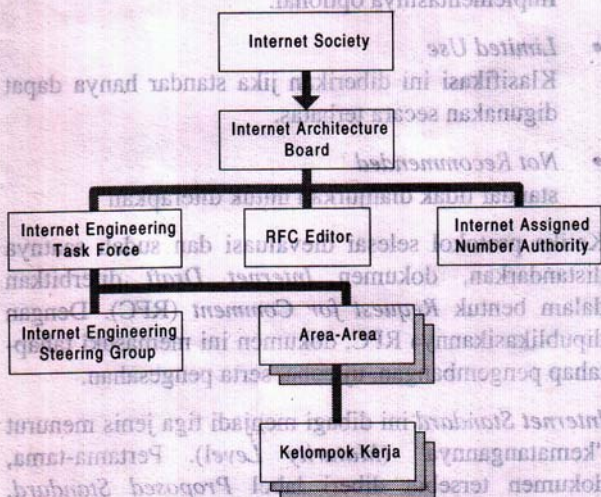
digunakan dalam protokol Internet (nomor port TCP, kode protokol IP, tipe hardware ARP dan lain-lain). Tugas ini didelegasikan ke lembaga yang disebut *Internet Assigned Numbers Authority* (IANA).

- *Internet Engineering Task Force* (IETF).ialah badan yang berorientasi untuk membentuk standar Internet. IETF ini dibagi menjadi sembilan kelompok kerja (misalnya aplikasi, routing dan addressing, keamanan komputer) dan bertugas menghasilkan standar-standar Internet. Untuk mengatur kerja IETF ini, dibentuk badan yang disebut *Internet Engineering Steering Group* (IESG)
- *Internet Research Task Force* (IRTF). Badan ini memiliki orientasi pada riset-riset jangka panjang.

Standar Internet datang dari badan kerja IETF. Badan ini terdiri atas pada peneliti dan insinyur yang sehari harinya bekerja di bidang pengembangan spesifikasi Internet. Tiap kelompok kerja (*working group*) memiliki bidang dan tanggung-jawab masing-masing. Hampir keseluruhan aktivitas kelompok ini dilakukan melalui e-mail. Akses terhadap aktivitas kelompok ini tersedia bagi setiap orang yang berminat. Dan karena tidak ada keanggotaan secara formal, cara untuk bergabung ke kelompok IETF ini ialah mengikuti mailing list kelompok tersebut.

1.2.2. Proses pembentukan standar Internet

Sebelum menjadi RFC, keseluruhan usulan spesifikasi protokol tersebut harus diterbitkan sebagai *Internet*

Draft. Dokumen ini bisa diterbitkan oleh siapa saja, baik itu badan kerja IETF, kelompok lain yang berminat, maupun individual. Oleh IETF, *Internet Draft* ini ditempatkan dalam beberapa FTP server di Internet, agar dapat diakses oleh pihak-pihak yang ingin melakukan evaluasi. Dokumen *Internet Draft* hanya memiliki umur enam bulan. Setelah umurnya habis, dokumen dihapus dari server.

Gambar 1.1 Bagan pengelola standar Internet

Dokumen *Internet Draft* dievaluasi oleh para ahli teknis Internet anggota IETF dan editor RFC, untuk kemudian diberi klasifikasi.

Ada lima jenis klasifikasi, yaitu:

- *Required*
  Internet draft dianggap sangat penting sehingga wajib diterapkan pada semua host dan router TCP/IP.

- *Recommended*
  semua host dan router dianjurkan untuk menerapkan standar tersebut

- *Elective*
  Implementasinya optional.

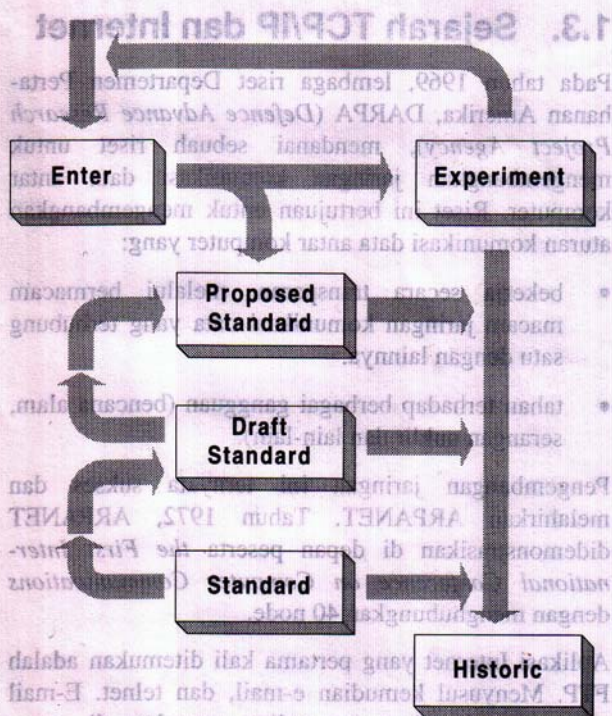
- *Limited Use*
  Klasifikasi ini diberikan jika standar hanya dapat digunakan secara terbatas.

- *Not Recommended*
  standar tidak dianjurkan untuk diterapkan

Ketika protokol selesai dievaluasi dan sudah saatnya distandarkan, dokumen *Internet Draft* diterbitkan dalam bentuk *Request for Comment* (RFC). Dengan dipublikasikannya RFC, dokumen ini memasuki tahap-tahap pengembangan, ujicoba, serta pengesahan.

Internet Standard ini dibagi menjadi tiga jenis menurut "kematangannya" (*Maturity Level*). Pertama-tama, dokumen tersebut diberi label *Proposed Standard*. Label ini menunjukkan bahwa dokumen di atas dianggap sudah lengkap secara teknis dan teoretis, namun belum cukup memiliki pengalaman operasional di lapangan.

Fase berikutnya ialah *Draft Standard*. Fase ini boleh dimasuki oleh suatu standar jika dalam berbagai macam implementasinya, masing-masing pihak telah menunjukkan interoperabilitas.

Langkah yang paling akhir ialah *Internet Standard*. *Internet Standard* ini ialah spesifikasi yang telah matang serta memiliki pengalaman operasional yang cukup signifikan.

Gambar 1.2 Alur pembentukan standar Internet

Tidak semua dokumen RFC berada dalam *Internet Standard Track*. Terdapat juga jalur lain, misalnya *Experimental*, *Informational*, atau *Historic*.

Isi RFC yang telah dipublikasikan tidak pernah direvisi. Jika protokol yang bersangkutan mengalami revisi, IETF akan menerbitkan RFC baru dan membuat RFC lama tidak berlaku.

1.3. Sejarah TCP/IP dan Internet

Pada tahun 1969, lembaga riset Departemen Pertahanan Amerika, DARPA (*Defence Advance Research Project Agency*), mendanai sebuah riset untuk mengembangkan jaringan komunikasi data antar komputer. Riset ini bertujuan untuk mengembangkan atùran komunikasi data antar komputer yang:

- bekerja secara transparan, melalui bermacam macam jaringan komunikasi data yang terhubung satu dengan lainnya.
- tahan terhadap berbagai gangguan (bencana alam, serangan nuklir dan lain-lain).

Pengembangan jaringan ini ternyata sukses dan melahirkan ARPANET. Tahun 1972, ARPANET didemonstrasikan di depan peserta *the First International Conference on Computer Communications* dengan menghubungkan 40 node.

Aplikasi Internet yang pertama kali ditemukan adalah FTP. Menyusul kemudian e-mail, dan telnet. E-mail menjadi aplikasi yang paling populer di masa ARPANET. Tahun 1979 tercatat sebagai tahun berdirinya USENET yang pada awalnya menghubungkan Universitas Duke dan UNC. Grup yang pertama kali dibentuk dalam USENET adalah grup net.*.

Ukuran ARPANET sendiri semakin lama semakin membesar. Protokol komunikasi data yang digunakan pada waktu itu, yaitu NCP (*Network Communication Protocol*), tidak sanggup menampung node komputer yang besar ini. DARPA kemudian mendanai pembuatan protokol komunikasi yang lebih umum. Protokol ini dinamakan TCP/IP. Departemen Pertahanan Amerika menyatakan TCP/IP menjadi standar untuk jaringannya pada 1982. Protokol ini kemudian diadopsi menjadi standar ARPANET pada tahun 1983. Perusahaan *Bolt Beranek Newman* (BBN) membuat protokol TCP/IP berjalan di atas komputer dengan sistem operasi UNIX. Pada saat itu lah dimulai perkawinan antara UNIX dan TCP/IP.

Pada tahun 1984 jumlah host di Internet melebihi 1000 buah. Pada tahun itu pula diperkenalkan *Domain Name System* (DNS) yang mengganti fungsi tabel nama host. Sistem domain inilah yang sampai saat ini kita gunakan untuk menuliskan nama host.

Tahun 1986, lembaga ilmu pengetahuan nasional Amerika Serikat U.S. *National Science Foundation* (NSF) mendanai pembuatan jaringan TCP/IP yang dinamai NSFNET. Jaringan ini digunakan untuk menghubungkan lima pusat komputer super dan memungkinkan terhubungnya universitas-universitas di Amerika Serikat dengan kecepatan jaringan tulang punggung sebesar 56kbps. Jaringan inilah yang kemudian menjadi embrio berkembangnya Internet yang kita kenal sekarang ini.

Pada tahun 1987 berdiri UUNET yang saat ini merupakan salah satu provider utama Internet. Tercatat pula pada tahun tersebut jumlah host melewati angka

10.000. Setahun kemudian kecepatan jaringan tulang punggung NSFNET ditingkatkan menjadi T1 (1,544 Mbps). Di samping itu juga terdapat beberapa negara di Eropa yang masuk ke jaringan NSFNET.

Perkembangan Internet menjadi semakin luas dan sampai menjangkau Australia dan Selandia Baru pada tahun 1989. Pada tahun tersebut jumlah host di Internet mencapai jumlah 100.000. Dua tahun kemudian aplikasi di Internet bertambah dengan diciptakannya *Wide Area Information Servers* (WAIS), *Gopher*, dan *World Wide Web* (WWW). Pada tahun tersebut kecepatan jaringan tulang punggung NFSNET ditingkatkan menjadi T3 (45Mbps).

Pada tahun 1992 jumlah host di Internet mencapai 1 juta host. Salah satu pemicu perkembangan ini adalah semakin meluasnya penggunaan layanan *Gopher* yang terdapat di Internet. Pada tahun ini juga untuk pertama kalinya dilaksanakan siaran audio dan video *multicast* melalui IETF MBONE (*multicast backbone*).

1.3.1. World Wide Web

Sebelum berkembangnya World Wide Web, Internet umumnya hanya digunakan oleh kalangan akademisi dan riset. Pada tahun 1993 NCSA mengeluarkan Mosaic, browser WWW dengan kemampuan grafik, pada seluruh platform yang biasa digunakan: X, PC/Windows, dan Macintosh. Munculnya Mosaic mulai menampakkan hasilnya pada tahun itu juga. Perkembangan lalu lintas data WWW tahun itu mencapai hampir 342000% sementara perkembangan Gopher 'hanya' 997%. Dunia bisnis dan media pun serta merta mulai memperhatikan Internet karena

perkembangan ini. Hadirnya Mosaic ternyata menjadi titik belok perkembangan Internet dari hanya digunakan oleh kalangan akademisi dan riset menjadi digunakan oleh orang banyak untuk bisnis dan hiburan.

Pada 17 November 1994 telah IETF menyetujui dokumen rekomendasi bagi protokol IP generasi baru yang disebut sebagai IPng menjadi usulan standar. IPng adalah IP versi baru yang dirancang sebagai langkah evolusi IP versi 4 yang sekarang ini digunakan. IPng dirancang untuk berjalan baik di jaringan berkecepatan tinggi (misal: ATM, OC-12, Gigabit Ethernet) maupun pada jaringan berkecepatan rendah (misal: jaringan nirkabel).

Dalam perkembangan Internet telah banyak muncul Penyedia Jasa Internet sehingga pada tahun 1995, NSFnet yang telah sekian lama menjadi tulang punggung Internet kembali menjadi jaringan untuk keperluan riset. Karena perubahan ini, lalu lintas data yang melalui Amerika dialihkan ke jaringan tulang punggung Penyedia Jasa Internet. Sementara itu NSFnet mengembangkan jaringan berkecepatan sangat tinggi yang menghubungkan lima pusat komputer super. Jaringan tersebut diberi nama *very high speed Backbone Network Service* (vBNS) dengan kecepatan 622Mbps (OC-12).

1.4. Layanan di TCP/IP (Internet Sekarang)

Sejak awal perkembangan TCP/IP dan Internet hingga sekarang, semakin banyak pula layanan yang diberikan oleh jaringan TCP/IP. Layanan-layanan TCP/IP dapat

diakses menggunakan program client yang spesifik untuk layanan tersebut atau menggunakan web browser untuk jenis layanan tertentu. Web browser menggunakan konsep URL (*Uniform Resource Locator*) untuk mengakses layanan tertentu pada jaringan TCP/IP dan Internet. Format URL adalah sebagai berikut:

<skema>:<bagian-skema-spesifik>

URL mengandung nama skema yang digunakan (<skema>) yang diikuti oleh tanda titik dua dan string (<bagian-skema-spesifik>) yang pengartiannya bergantung pada skema yang digunakan. Contoh URL, misalnya:

ftp://ftp.cdrom.com/pub/

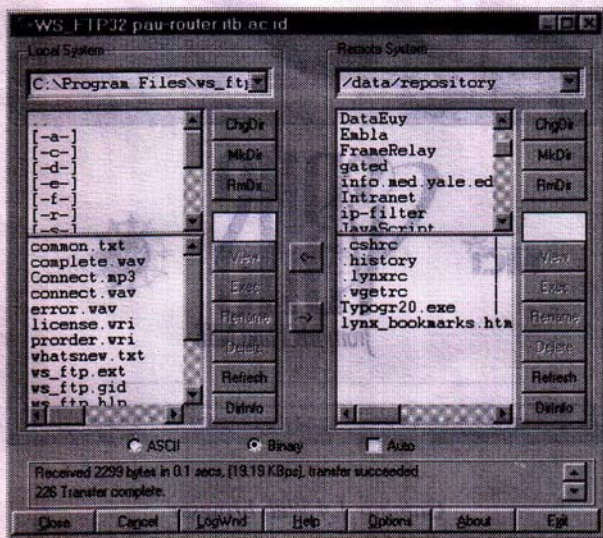
http://www.yahoo.com/Arts/Humanities/

mailto:cnrg@itb.ac.id

nntp://rain.psg.com/

file:///D|/CNRG Data/Netscape/REC-html32.html

Dari contoh-contoh URL di atas terlihat skema untuk mengakses resource di jaringan TCP/IP. URL ftp://ftp.cdrom.com/pub/ menyatakan akses layanan FTP di host ftp://ftp.cdrom.com pada direktori /pub/. URL mailto:cargo@itb.ac.id untuk mengirim e-mail kepada mailto:cnrg@itb.ac.id

Layanan yang diciptakan pada awal perkembangan TCP/IP adalah FTP (File Transfer Protocol). Dengan protokol ini, komputer-komputer dapat saling mengirim file. Pengiriman file ini dapat dilakukan dalam mode ASCII untuk file-file teks atau mode *binary* untuk file-file dengan tipe *byte-stream*, misal: file gambar.

Gambar 1.3 Layanan transfer file

Layanan World Wide Web (WWW) saat ini adalah layanan yang paling populer di antara seluruh jenis layanan TCP/IP. Server WWW diakses dengan menggunakan WWW browser seperti Netscape dan Internet Explorer. Protokol yang digunakan untuk layanan WWW ini adalah HTTP (*Hypertext Transfer Protocol*). Versi HTTP yang digunakan di Internet sekarang ini adalah versi 1.0 dan 1.1. HTTP versi 1.1 adalah evolusi dari versi 1.0 yang akan mengatasi kelemahan-kelemahan yang terdapat pada HTTP versi 1.0.

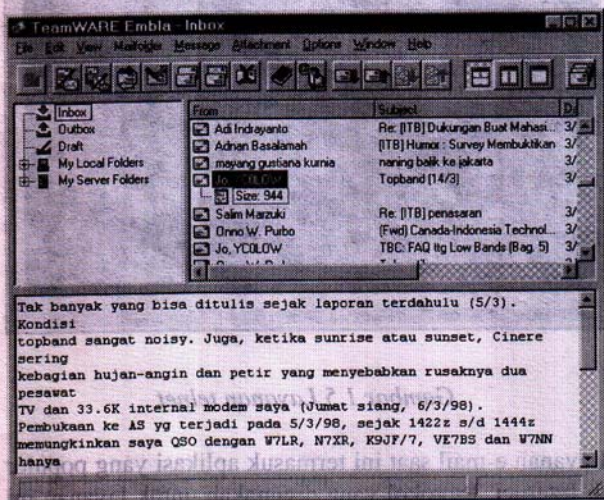
Gambar 1.4 Web browser

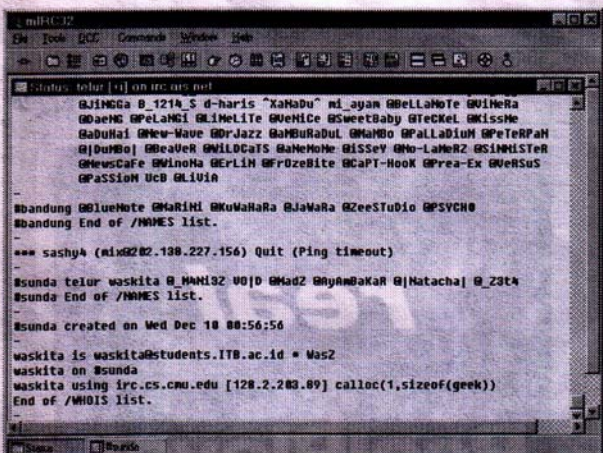
Layanan telnet memungkinkan pengguna Internet untuk masuk ke komputer lain dan menjalankan perintah-perintah di komputer tersebut. Layanan ini banyak diimplementasikan pada sistem operasi Unix. Windows NT 4.0 tidak menyediakan jenis layanan ini sedangkan Novell IntranetWare mendukung jenis layanan telnet untuk mengakses konsol server IntranetWare dari jarak jauh.

Gambar 1.5 Layanan telnet

Layanan e-mail saat ini termasuk aplikasi yang populer di Internet. Protokol yang digunakan untuk layanan ini adalah SMTP (*Simple Mail Transport Protocol*) untuk pengiriman e-mail, POP (Post Office Protocol) dan IMAP (*Internet Message Access Protocol*) untuk meng-ambil e-mail dari server. E-mail dapat juga mengirimkan data selain teks biasa dengan menggunakan MIME (*Multipurpose Internet Mail Extensions*).

Layanan USENET adalah jenis layanan yang mirip dengan e-mail. Dalam layanan news, pengguna mengirimkan surat ke newsgroup yang mendiskusikan topik-topik tertentu. Server-server di USENET saling terhubung dan meneruskan setiap surat yang diterimanya ke server lain berdasarkan aturan yang telah disepakati antar server. Protokol untuk mendistribusikan news adalah NNTP (*Network News Transport*

Protocol). Layanan e-mail dan news termasuk dalam kategori Internet Text Message, jadi terdapat kemiripan format teks pada keduanya.

Gambar 1.6 Layanan e-mail

Layanan IRC (Internet Relay Chat) termasuk salah satu layanan interaktif yang dapat ditemukan pada jaringan TCP/IP. IRC memberikan layanan *chat* bagi pengguna jaringan TCP/IP. Di Internet terdapat banyak server IRC yang saling berhubungan dan pengguna Internet cukup masuk ke salah satu server untuk dapat 'chatting' dengan pengguna lain walaupun mereka masuk ke server yang berbeda.

Gambar 1.7 Internet Relay Chat

Perkembangan layanan di Internet bertambah dengan adanya layanan audio dan video yang bersifat streaming. Streaming adalah sebuah jenis layanan yang langsung mengolah data yang diterima tanpa menunggu seluruh data selesai dikirim. Layanan yang bersifat streaming saat ini adalah layanan audio dan video. Data audio dan video biasanya berukuran sangat besar. Untuk menampilkan video selama satu menit ukuran file-nya dapat mencapai 1Mbyte. Karena mengambil data sebesar itu dapat memerlukan waktu yang lebih lama daripada memainkannya maka digunakan layanan yang bersifat streaming. Contoh aplikasi yang menggunakan layanan streaming adalah RealPlayer dari RealNetworks.

Gambar 1.8 RealPlayer

Di samping layanan-layanan yang disebutkan di atas, tentu banyak lagi layanan yang terdapat di jaringan TCP/IP (Internet) saat ini. Melihat tren perkembangan layanan di Internet, sepertinya kita akan menemukan lebih banyak lagi layanan multimedia dan hiburan di masa mendatang.

1.5. Mengenai Buku Ini

Buku ini pada dasarnya memberikan penjelasan teknis mengenai protokol yang digunakan di Internet yaitu TCP/IP. Penjelasan di buku ini umumnya bersifat tidak tergantung terhadap sistem operasi dan perangkat keras, kecuali pada contoh-contoh yang diberikan. Contoh-contoh dalam buku ini menggunakan sistem operasi Unix FreeBSD dan Linux karena sistem operasi ini dapat diperoleh dengan gratis dan mendukung protokol TCP/IP dengan baik.

Buku dimulai dengan penjelasan mengenai konsep dasar jaringan TCP/IP yang menjelaskan mengenai dasar-dasar arsitektur TCP/IP yang menghubungkan Internet. Penjelasan kemudian dilanjutkan dengan Internet Protocol, protokol utama dalam arsitektur TCP/IP yang menangani pengalamatan host-host dalam jaringan serta penyampaian data antar host tersebut.

Bab 4 membahas mengenai sistem penamaan host-host di jaringan TCP/IP, yaitu *Domain Name System* (DNS). Pada bab ini juga dijelaskan bagaimana mengatur konfigurasi DNS menggunakan *Berkeley Internet Name Daemon* (BIND).

Proses routing yang bertujuan untuk menyampaikan data antar host dibahas lebih dalam di bab selanjutnya. Di samping penjelasan mengenai standar yang digunakan di Internet, diberikan pula contoh implementasi routing di FreeBSD menggunakan GateD di Bab 6.

Bab 7 menjelaskan protokol aplikasi di TCP/IP yang digunakan untuk mengatur jaringan. Protokol ini disebut sebagai *Simple Network Management Protocol* (SNMP). Berbagai parameter jaringan TCP/IP yang

dapat diatur juga dijelaskan di sini. Contoh implementasi SNMP di bab ini menggunakan UCD-snmpd yang dijalankan di Linux.

Bab 8, bab terakhir, membicarakan tiga buah protokol aplikasi yang populer saat ini, yaitu, File Transfer Protocol (FTP), Simple Mail Transport Protocol (SMTP), dan Hypertext Transfer Protocol (HTTP)

Bab 2
Konsep Dasar TCP/IP

Pada bab ini akan diberikan pengantar tentang konsep dasar dan cara kerja Protokol TCP/IP yang menjadi dasar bagi terbentuknya jaringan internet. Melalui pengantar ini pembaca akan memperoleh dasar yang kuat untuk memahami bab-bab selanjutnya.

2.1. Dasar Arsitektur TCP/IP

Pada dasarnya, komunikasi data merupakan proses mengirimkan data dari satu komputer ke komputer yang lain. Untuk dapat mengirimkan data, pada komputer harus ditambahkan alat khusus, yang dikenal sebagai network interface (interface jaringan). Jenis interface jaringan ini bermacam-macam, bergantung pada media fisik yang digunakan untuk mentransfer data tersebut.

Dalam proses pengiriman data ini terdapat beberapa masalah yang harus dipecahkan. Pertama, data harus dapat dikirimkan ke komputer yang tepat, sesuai tujuannya. Hal ini akan menjadi rumit jika komputer tujuan transfer data ini tidak berada pada jaringan lokal, melainkan di tempat yang jauh. Jika lokasi komputer yang saling berkomunikasi "jauh" (secara jaringan) maka terdapat kemungkinan data rusak atau

hilang. Karenanya, perlu ada mekanisme yang mencegah rusaknya data ini.

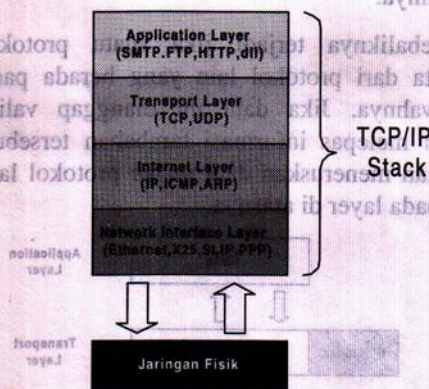
Hal lain yang perlu diperhatikan ialah, pada komputer tujuan transfer data mungkin terdapat lebih dari satu aplikasi yang menunggu datangnya data. Data yang dikirim harus sampai ke aplikasi yang tepat, pada komputer yang tepat, tanpa kesalahan.

Cara alamiah untuk menghadapi setiap masalah yang rumit ialah memecahkan masalah tersebut menjadi bagian yang lebih kecil. Dalam memecahkan masalah transfer data di atas, para ahli jaringan komputer pun melakukan hal yang sama. Untuk setiap problem komunikasi data, diciptakan solusi khusus berupa aturan-aturan untuk menangani problem tersebut. Untuk menangani semua masalah komunikasi data, keseluruhan aturan ini harus bekerja sama satu dengan lainnya. Sekumpulan aturan untuk mengatur proses pengiriman data ini disebut sebagai *protokol komunikasi data*. Protokol ini diimplementasikan dalam bentuk program komputer (software) yang terdapat pada komputer dan peralatan komunikasi data lainnya.

TCP/IP adalah sekumpulan protokol yang didesain untuk melakukan fungsi-fungi komunikasi data pada *Wide Area Network* (WAN). TCP/IP terdiri atas sekumpulan protokol yang masing-masing bertanggung jawab atas bagian-bagian tertentu dari komunikasi data. Berkat prinsip ini, tugas masing-masing protokol menjadi jelas dan sederhana. Protokol yang satu tidak perlu mengetahui cara kerja protokol yang lain, sepanjang ia masih bisa saling mengirim dan menerima data.

Berkat penggunaan prinsip ini, TCP/IP menjadi protokol komunikasi data yang fleksibel. Protokol TCP/IP

dapat diterapkan dengan mudah di setiap jenis komputer dan interface jaringan, karena sebagian besar isi kumpulan protokol ini tidak spesifik terhadap satu komputer atau peralatan jaringan tertentu. Agar TCP/IP dapat berjalan di atas interface jaringan tertentu, hanya perlu dilakukan perubahan pada protokol yang berhubungan dengan interface jaringan saja.

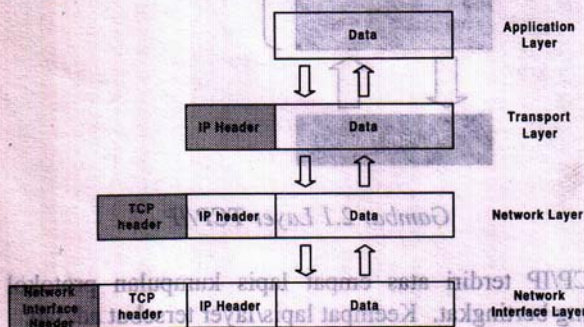
Sekumpulan protokol TCP/IP ini dimodelkan dengan empat layer TCP/IP, sebagaimana terlihat pada gambar di bawah ini.

Gambar 2.1 Layer TCP/IP

TCP/IP terdiri atas empat lapis kumpulan protokol yang bertingkat. Keempat lapis/layer tersebut adalah:

- Network Interface Layer
- Internet Layer
- Transport Layer
- Application Layer

Dalam TCP/IP, terjadi penyampaian data dari protokol yang berada di satu layer ke protokol yang berada di layer yang lain. Setiap protokol memperlakukan semua informasi yang diterimanya dari protokol lain sebagai data.

Jika suatu protokol menerima data dari protokol lain di layer atasnya, ia akan menambahkan informasi tambahan miliknya ke data tersebut. Informasi ini memiliki fungsi yang sesuai dengan fungsi protokol tersebut. Setelah itu, data ini diteruskan lagi ke protokol pada layer di bawahnya.

Hal yang sebaliknya terjadi jika suatu protokol menerima data dari protokol lain yang berada pada layer di bawahnya. Jika data ini dianggap valid, protokol akan melepas informasi tambahan tersebut, untuk kemudian meneruskan data itu ke protokol lain yang berada pada layer di atasnya.

Gambar 2.2 Pergerakan data dalam layer TCP/IP

Lapisan/Layer terbawah, yaitu *Network Interface layer*, bertanggung jawab mengirim dan menerima data ke

dan dari media fisik. Media fisiknya dapat berupa kabel, serat optik, atau gelombang radio. Karena tugasnya ini, protokol pada layer ini harus mampu menerjemahkan sinyal listrik menjadi data digital yang dimengerti komputer, yang berasal dari peralatan lain yang sejenis.

Lapisan/Layer protokol berikutnya ialah *Internet Layer*. Protokol yang berada pada layer ini bertanggung jawab dalam proses pengiriman paket ke alamat yang tepat. Pada layer ini terdapat tiga macam protokol, yaitu IP, ARP, dan ICMP.

IP (Internet Protocol) berfungsi untuk menyampaikan paket data ke alamat yang tepat. ARP (*Address Resolution Protocol*) ialah protokol yang digunakan untuk menemukan alamat hardware dari host/komputer yang terletak pada network yang sama. Sedangkan ICMP (Internet Control Message Protocol) ialah protokol yang digunakan untuk mengirimkan pesan dan melaporkan kegagalan pengiriman data.

Layer berikutnya, yaitu *Transport Layer*, berisi protokol yang bertanggung jawab untuk mengadakan komunikasi antara dua host/komputer. Kedua protokol tersebut ialah TCP (*Transmission Control Protocol*) dan UDP (User Datagram Protocol).

Layer teratas, ialah *Application Layer*. Pada layer inilah terletak semua aplikasi yang menggunakan protokol TCP/IP ini.

2.2. Analogi Pengiriman Surat

Fungsi masing-masing layer/lapisan protokol serta aliran data pada layer TCP/IP di atas, dapat secara mudah diterangkan dengan menggunakan analogi-analogi yang sederhana.

Bayangkan diri Anda adalah seorang yang tinggal di kota tertentu yang hendak mengirim surat ke saudara Anda di rumahnya, di kota lain. Apa sajakah yang Anda lakukan?

Tentu saja Anda mula-mula menulis dulu isi surat tersebut. Anda mengambil selembar kertas, dan menggunakan ballpoint untuk menulis berita yang hendak Anda sampaikan ke saudara Anda di rumah.

Setelah Anda selesai menulis surat, Anda akan mengambil amplop dan memasukkan surat tersebut kedalam amplop.

Mengapa surat tersebut harus dimasukkan ke dalam amplop? Tentu saja agar ia terlindung dari hal-hal yang bisa merusaknya. Dengan menggunakan amplop surat mempunyai kemungkinan rusak lebih kecil dibandingkan dengan tanpa menggunakan amplop.

Nah, saat Anda hendak menggunakan amplop untuk mengirim surat tersebut. Anda dihadapkan pada dua pilihan, yaitu apakah Anda menggunakan amplop tertutup ataukah amplop terbuka.

Jika Anda menggunakan amplop tertutup, surat Anda lebih terlindungi dibandingkan dengan menggunakan amplop terbuka. Namun konsekuensinya, dengan amplop terbuka Anda dapat menggunakan prangko yang

lebih murah untuk mengirimkan surat Anda ke saudara Anda di rumah.

Tentu saja, jika isi suratnya sangat penting, Anda bukan saja menggunakan amplop tertutup. Bila perlu, Anda akan menggunakan jasa pos tercatat, agar kemungkinan hilangnya surat Anda menjadi kecil.

Nah, setelah surat ditulis, dimasukkan amplop, dan diberi perangko, apakah surat tersebut dapat segera dikirimkan? Tentu saja tidak. Anda harus terlebih dahulu menuliskan kepada siapa surat tersebut hendak dikirimkan: nama saudara Anda tersebut. Hal ini menjadi penting karena Anda ternyata adalah satu dari tujuh bersaudara. Dan pada alamat yang Anda tuju, rumah saudara Anda tersebut, tinggal pula 5 saudara Anda yang lain. Selain itu, di bagian belakang amplop surat, biasanya kita tuliskan nama pengirim dan alamat pengirim surat, yaitu nama dan alamat kita sendiri. Hal ini penting karena bisa jadi saudara Anda tersebut akan membalas surat Anda dan dia tentu harus mengetahui alamat Anda.

Setelah menulis nama tujuan surat itu, Anda tentu harus menuliskan alamat saudara Anda tersebut. Tanpa alamat yang jelas, besar kemungkinan surat itu tidak akan sampai ke tempat tujuannya. Format alamatnya pun harus jelas. Anda harus menuliskan nama jalan, nomer rumah, kode pos, kota, bahkan propinsi dan negara tempat tinggal saudara Anda. Jika alamatnya jelas (dan prangkonya cukup), surat pasti akan sampai ke tempat tujuan, kecuali terdapat ada dua rumah yang mempunyai alamat yang sama (suatu hal yang tidak boleh terjadi).

Setelah semua proses di atas selesai, barulah Anda dapat mengirimkan surat Anda. Anda bisa datang ke kantor pos, atau pun memasukkan surat Anda ke Bis Surat di pinggir jalan. Tugas Anda selesai, dan Perusahaan Pos akan mengurus selebihnya.

Cara kerja protokol TCP/IP dalam satu komputer sangatlah mirip dengan cerita di atas.

Ketika Anda mengirimkan e-mail melalui program aplikasi di PC Anda (misal: Eudora, Netscape Mail, atau Internet Explorer), e-mail Anda terlebih dahulu diolah oleh protokol TCP (Transmission Control Protocol). Saat ia diolah, protokol TCP ini memberikan ***amplop*** untuk melindungi data yang hendak Anda kirim, yang berupa data tambahan. Data tambahan ini antara lain berupa *Sequence Number* (nomer urut data), *Acknowledgement Number*, dan *16 bit checksum* untuk pemeriksaan kesalahan data.

Selain data di atas, pada "*amplop*" TCP juga ditambahkan data berupa *16 bit source port number* dan *16 bit destination port number*. Kedua hal ini bisa dianalogikan dengan nama pengirim surat dan nama penerima surat pada cerita di atas. *Destination port number* diperlukan karena pada komputer tujuan bisa jadi terdapat banyak aplikasi TCP/IP yang siap menerima data. Data yang kita kirim harus sampai ke aplikasi yang tepat. Yaitu aplikasi yang berhak menerimanya. Hal ini bisa dianalogikan dengan enam saudara Anda lainnya yang tinggal serumah, pada cerita di atas.

Pada pengiriman surat di atas, Anda tahu siapa yang dituju karena kebetulan saja dia adalah saudara Anda sendiri. Ada saat dimana Anda harus mengirimkan surat ke instansi tertentu untuk keperluan tertentu. Dan

pada saat itu Anda tetap tahu siapa yang harus dituju, walaupun Anda sama sekali tidak pernah menginjakkan kaki di instansi tersebut.

Jika Anda hendak mengirimkan surat lamaran pekerjaan ke suatu perusahaan, hampir dapat dipastikan Anda akan mengirimnya ke bagian HRD atau personalia perusahaan tersebut. Jika Anda melihat ketidakberesan di instansi Anda, Anda mengirimkan surat ke kotak pos 5000. Mengapa hal ini bisa dilakukan? Karena ia sudah diketahui secara umum.

Pada TCP pun terjadi hal yang sama. Jika saat komputer hendak melakukan transaksi TCP dengan komputer lain, ia harus terlebih dahulu mengetahui port number manakah yang hendak ia hubungi untuk keperluan tertentu. Untuk setiap aplikasi TCP di internet telah distandardkan *port number* tertentu.

Misalkan, jika kita hendak mengirimkan email ke komputer tertentu, dengan protokol SMTP, protokol TCP di komputer kita harus menghubungi protokol TCP port 25 di komputer lawan. Jika kita hendak melakukan transfer file dengan protokol FTP, yang harus di hubungi ialah port 21. Jika kita hendak mengambil data halaman web, biasanya yang di kontak ialah port 80. Angka angka ini telah distandardkan pada protokol TCP, dan dikenal sebagai *Well Known Port*. Standarisasi ini diatur oleh lembaga yang dinamakan Internet Assigned Number Authority. ***(http:// www.isi.edu/iana/)***

Selain TCP, terdapat pula protokol komunikasi data yang setingkat dengannya, yaitu UDP (User Datagram Protocol). Paket UDP berbeda dengan TCP dalam hal

reliabilitas/keandalan. Analogi TCP dan UDP mirip dengan surat dengan amplop terbuka dan tertutup.

Dengan menggunakan TCP, keandalan pengiriman data terjamin karena pada TCP terdapat proses data acknowledgement, retransmisi dan sequencing (pengurutan). Dengan menggunakan dua proses ini, TCP selalu meminta konfirmasi setiap kali selesai mengirim data, apakah data telah sampai dengan selamat di tempat tujuan. Jika data berhasil mencapai tujuan, TCP akan mengirimkan data urutan berikutnya. Jika tidak, TCP akan melakukan retransmisi (pemancaran ulang data tersebut). Data yang dikirim dan diterima pun diatur berdasarkan nomer urut (Sequence Number)

Pada paket UDP, dengan tidak adanya *field sequence number* dan *acknowledge number*, hal ini tidak dilakukan. Akibatnya, Protokol layer atas yang menggunakan UDP tidak pernah mengetahui sampai tidaknya paket yang dikirimnya sampai ke tempat tujuan.

Namun terkadang ada aplikasi yang tidak membutuhkan keamanan data seketat TCP. Untuk data data tertentu yang dipancarkan secara periodik dan berukuran kecil, ke tempat yang tidak jauh, UDP tetap digunakan, dan dapat berperan lebih efisien. Dalam analogi surat, Anda bisa menggunakan prangko yang lebih murah dengan protokol *amplop terbuka* ini.

Setelah data diproses oleh protokol TCP agar terjamin keutuhannya , data ini diteruskan ke protokol di bawah TCP, yaitu IP (Internet Protocol). IP adalah protokol di Internet yang mengurusi masalah pengalamatan dan mengatur pengiriman paket data sehingga ia sampai ke alamat yang benar.

Agar kita dapat mengirimkan surat ke rumah tertentu, rumah itu harus memiliki alamat. Hal yang sama terjadi pada dunia internet. Setiap komputer yang terkoneksi ke internet harus memiliki alamat. Alamat ini harus unik. **Satu alamat hanya boleh dimiliki oleh satu komputer.** Sebagaimana satu alamat rumah hanya boleh dimiliki oleh satu rumah. Bayangkan betapa bingungnya tukang pos jika dalam satu kota terdapat dua jalan dengan nama yang sama. Karena alasan itulah kita menggunakan kode pos untuk membantu proses pengalamatan.

Alamat komputer dalam internet ini disebut sebagai *IP address*. IP address biasanya ditulis sebagai 4 urutan bilangan desimal yang dipisahkan dengan titik. Setiap bilangan tersebut berupa salah satu bilangan yang berharga diantara 0-255 (nilai desimal yang mungkin untuk 1 byte). Contoh penulisan IP address ialah sebagai berikut.

132.92.122.114

Secara teoretis, dengan menggunakan format seperti di atas, jumlah IP address yang tersedia ialah 255 x 255 x 255 x 255 IP address.

Siapa yang mengatur pengalokasian IP address? siapa yang harus kita mintai IP address jika kita membutuhkannya? pengalokasian IP address ini diatur oleh lembaga yang disebut Internic (Internet Network Information Center) ***(http://www.internic.net)***

Keterangan lebih lanjut tentang format dan pengalokasian IP address ini dapat dilihat pada bab berikutnya, Bab 3, tentang IP address.

Paket IP terdiri atas paket yang diterimanya dari TCP, ditambah dengan beberapa data tambahan, diantaranya ialah 32 bit *source IP address* (IP address asal), 32 bit *destination IP address* (IP address tujuan).

Setelah paket data diproses menjadi paket IP, kemanakah paket ini dikirim? Sebagaimana Anda memasukkan surat Anda ke bis surat, paket data ini dikirimkan ke protokol yang berada di Network Interface Layer.

Network Interface Layer ialah bagian/lapisan komunikasi data yang berfungsi untuk mengatur akses data ke medium fisik. Layer inilah yang mengatur pengiriman dan pengambilan data dari media fisik.

2.3. Komponen Fisik dalam Jaringan TCP/IP

Komputer dengan protokol TCP/IP dapat terhubung ke komputer lain dan jaringan lain karena bantuan peralatan jaringan komputer. Pada komputer itu sendiri, ditambahkan alat yang disebut *network interface*. Network interface ini bisa berupa *card ethernet* atau *modem*. Card ethernet terhubung atau komputer lain via kabel RG-58 atau ke hub ethernet via kabel UTP. Modem terhubung ke jaringan melalui kabel telepon. Diluar peralatan yang disebutkan ini, masih diperlukan peralatan lain untuk membentuk jaringan komputer. Peralatan ini disebut sebagai Device Penghubung Jaringan .

Device penghubung jaringan ini secara umum dibagi dalam beberapa katagori:

- Repeater
- Bridge
- Router

2.3.1. Repeater

Fasilitas paling sederhana dalam jaringan komputer adalah *repeater*. Fungsi utama *repeater* adalah menerima sinyal dari satu segmen kabel LAN dan memancarkannya kembali dengan kekuatan yang sama dengan sinyal asli pada segmen (satu atau lebih) kabel LAN yang lain. Dengan adanya repeater ini, jarak antara dua jaringan komputer bisa diperjauh.

2.3.2. Bridge

Sebuah bridge juga meneruskan paket dari satu segmen LAN ke segmen lain, tetapi bridge lebih fleksibel dan lebih cerdas dibandingkan dengan repeater. Bridge berkerja dengan meneruskan paket ethernet dari satu jaringan ke jaringan lain. Tiap card ethernet memiliki alamat ethernet (*ethernet address*) yang unik. Beberapa bridge mempelajari alamat ethernet setiap device yang terhubung dengannya dan mengatur alur frame berdasarkan alamat tersebut.

Bridge dapat menghubungkan jaringan yang menggunakan metode transmisi berbeda dan/atau medium access control yang berbeda. Misalnya, bridge dapat menghubungkan Ethernet baseband dengan Ethernet broadband. Bridge mungkin juga menghubungkan LAN Ethernet dengan LAN token ring, untuk fungsi

ini, bridge harus mampu mengatasi perbedaan format paket setiap frame di atas.

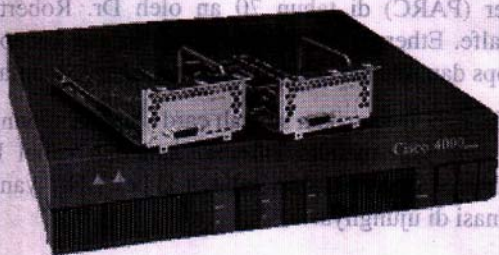
Bridge mampu memisahkan sebagian trafik karena mengimplementasikan mekanisme pemfilteran frame (*frame filtering*). Mekanisme yang digunakan di bridge ini umum disebut sebagai *store and forward* sebab frame yang diterima disimpan sementara di bridge dan kemudian di-forward ke worksation di LAN lain. Walaupun demikian, *broadcast traffic* yang dibangkitkan dalam LAN tidak dapat difilter oleh bridge.

2.3.3. Router

Router memiliki kemampuan melewatkan paket IP dari satu jaringan ke jaringan lain yang mungkin memiliki banyak jalur di antara keduanya. Router-router yang saling terhubung dalam jaringan internet turut serta dalam sebuah algoritma routing terdistribusi untuk menentukan jalur terbaik yang dilalui paket IP dari satu sistem ke sistem lain.

Router dapat digunakan untuk menghubungkan sejumlah LAN (*Local Area Network*) sehingga trafik yang dibangkitkan oleh suatu LAN terisolasikan dengan baik dari trafik yang dibangkitkan oleh LAN lain. Jika dua atau lebih LAN terhubung dengan router, setiap LAN dianggap sebagai subnetwork yang berbeda. Mirip dengan bridge, router dapat menghubungkan network interface yang berbeda.

Router yang umum dipakai terdiri atas dua jenis, yaitu router dedicated (buatan pabrik, misalnya Cisco http://www.cisco.com, BayNetworks http://www.baynetworks.com) dan PC router. PC dapat difungsikan

sebagai router sepanjang ia memiliki lebih dari satu interface jaringan, mampu memforward paket IP, serta menjalankan program untuk mengatur routing paket.

Gambar 2.3 Router buatan pabrik

2.4. Protokol-Protokol dalam TCP/IP

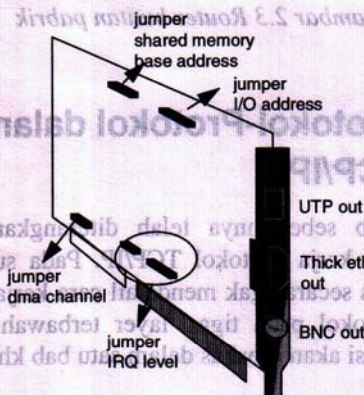
Pada subbab sebelumnya telah diterangkan sekilas tentang cara kerja protokol TCP/IP. Pada subbab ini akan dibahas secara agak mendetail cara kerja masing-masing protokol pada tiga layer terbawah TCP/IP. Layer aplikasi akan dibahas dalam satu bab khusus.

2.4.1. Network Interface Layer

Layer ini bertanggung jawab mengirim data dan menerima data dari media fisik. Beberapa contohnya ialah ethernet, SLIP dan PPP.

2.4.2. Ethernet

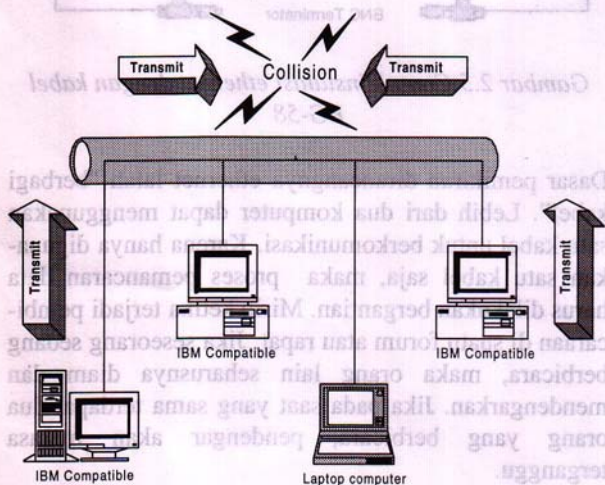
Jika Anda mengenal Local Area Network, Anda dapat dipastikan mengenal Interface Ethernet. Model interface Ethernet ditemukan di Xerox Palo Alto Research Center (PARC) di tahun 70 an oleh Dr. Robert M. Metcalfe. Ethernet pertama berjalan dengan kecepatan 3-Mbps dan dikenal sebagai Ethernet Eksperimental.

Interface ini merupakan sebuah card yang terhubung ke card yang lain melalui *ethernet hub* dan kabel UTP atau hanya menggunakan sebuah kabel BNC yang di terminasi di ujungnya.

Gambar 2.4 Model card ethernet

Gambar 2.5 Contoh instalasi ethernet dengan kabel RG-58

Dasar pemikiran dirancangnya ethernet ialah "berbagi kabel”. Lebih dari dua komputer dapat menggunakan satu kabel untuk berkomunikasi. Karena hanya digunakan satu kabel saja, maka proses pemancaran data harus dilakukan bergantian. Mirip ketika terjadi pembicaraan di suatu forum atau rapat. Jika seseorang sedang berbicara, maka orang lain seharusnya diam dan mendengarkan. Jika pada saat yang sama terdapat dua orang yang berbicara, pendengar akan merasa terganggu.

Sebelum satu card ethernet memancarkan datanya pada kabel, dia harus mendeteksi terlebih dahulu ada tidaknya card lain yang sedang memancar. Jika tidak ada, maka dia akan memancar. Jika ada, maka card ethernet yang bersangkutan akan menunggu sampai kabel dalam keadaan *kosong*.

Jika pada saat yang bersamaan, dua card memancarkan data maka terjadilah collision/tabrakan (hal ini dideteksi oleh card yang bersangkutan dengan memeriksa tegangan kabel, jika tegangan ini melampaui batas tertentu, maka telah terjadi *collision*). Jika *collision* terjadi, maka masing-masing card ethernet berhenti memancar dan menunggu lagi dengan ***selang waktu yang acak*** untuk mencoba memancar kembali. Karena selang waktu pancar masing-masing card yang acak ini, maka kemungkinan *collision* lebih lanjut menjadi lebih kecil.

Gambar 2.6 Collision ethernet

Seluruh proses di atas, dikenal dengan nama CSMA/CD (Carrier Sense Multiple Access/Collision Detection)

Karena dalam satu kabel terdapat banyak card ethernet, maka harus ada suatu metode untuk mengenali dan membedakan masing-masing card ethernet tersebut. Untuk itu, pada setiap card ethernet telah tertera kode khusus sepanjang 48 bit, yang dikenal sebagai *ethernet address*.

2.4.3. SLIP & PPP

Selain ethernet, interface jaringan yang sangat banyak dipakai ialah modem telepon, yang dihubungkan ke komputer via serial port. Protokol yang banyak dipakai untuk menangani jalur serial ini ialah SLIP (*Serial Line Interface Protocol*) dan PPP (*Point to Point Protocol*)

Serial Line Internet Protocol (SLIP)

SLIP ialah teknik enkapsulasi datagram yang paling sederhana di internet. Datagram IP yang diterima dienkapsulasi dengan menambahkan karakter END (0xC0) pada awal dan akhir frame. Jika pada datagram terdapat karakter 0xC0, karakter ini diterjemahkan sebagai karakter SLIP ESC, yaitu 0xDB 0xDC. Jika pada datagram sudah terdapat karakter 0xDB, karakter ini diubah menjadi 0xDB 0xDD.

Point to Point Protocol (PPP)

PPP terdiri atas beberapa protokol mini. Protokol tersebut adalah sebagai berikut:

- LCP (*Link Control Protocol*). LCP ini berfungsi membentuk dan memelihara link.

- *Authentication Protocol.* Protokol ini digunakan untuk memeriksa boleh tidaknya user menggunakan link ini. Ada dua jenis autentikasi yang umum digunakan, yaitu *Password Authentication Protocol* (PAP) dan *Challenge Handshake Authentication Protocol* (CHAP).

- *Network Control Protocol* (NCP). NCP berfungsi mengkoordinasi operasi bermacam-macam protokol jaringan yang melalui link PPP ini. Beberapa hal yang dilakukan oleh protokol ini ialah menegosiasikan jenis protokol kompresi yang akan dipakai serta menanyakan IP address mitranya.

2.4.4. Internet Layer

IP (Internet Protocol)

Protokol IP merupakan inti dari protokol TCP/IP. Seluruh data yang berasal dari protokol pada layer di atas IP harus dilewatkan, diolah oleh protokol IP, dan dipancarkan sebagai paket IP, agar sampai ke tujuan. Dalam melakukan pengiriman data, IP memiliki sifat yang dikenal sebagai *unreliable, conectionless, datagram delivery service*.

Ada dua kata yang menarik dari sifat IP di atas, yaitu *unreliable* dan *connectionless.*

Unreliable/Ketidakandalan berarti bahwa Protokol IP tidak menjamin datagram yang dikirim pasti sampai ke tempat tujuan. Protokol IP hanya berjanji ia akan melakukan usaha sebaik baiknya (*best effort delivery service*), agar paket yang dikirim tersebut sampai ke tujuan.

Gambar 2.8 Perjalanan IP. Paket 3,4,5 melalui jalur berbeda dengan 1 dan 2. Paket 3 ditransmisikan ulang karena router 2 down, sehingga kedatangannya tak urut seperti semula

Mengapa metode seperti ini dipakai dalam pengiriman paket IP? Hal ini dilakukan untuk menjamin tetap sampainya paket IP ini ke tujuan, walaupun salah satu jalur ke tujuan itu mengalami masalah.

Version	Header Length	Type of Service		Total Length of Datagram
Identification			Flags	Fragment Offset
Time to Live		Protocol	Header Checksum	
Source IP Address				
Destination IP Address				
Options Strict Source Routing, Loose Souce Routing				
DATA				

Gambar 2.9 Format datagram IP

Gambar 2.8 Perjalanan IP. Paket 3,4,5 melalui jalur berbeda dengan 1 dan 2. Paket 3 ditransmisikan ulang karena router 2 down, sehingga kedatangannya tak urut seperti semula

Mengapa metode seperti ini dipakai dalam pengiriman paket IP? Hal ini dilakukan untuk menjamin tetap sampainya paket IP ini ke tujuan, walaupun *salah satu jalur* ke tujuan itu mengalami masalah.

<table>
<tr><td>Version</td><td>Header Length</td><td>Type of Service</td><td colspan="2">Total Length of Datagram</td></tr>
<tr><td colspan="3">Identification</td><td>Flags</td><td>Fragment Offset</td></tr>
<tr><td colspan="2">Time to Live</td><td>Protocol</td><td colspan="2">Header Checksum</td></tr>
<tr><td colspan="5">Source IP Address</td></tr>
<tr><td colspan="5">Destination IP Address</td></tr>
<tr><td colspan="5">Options
Strict Source Routing, Loose Source Routing</td></tr>
<tr><td colspan="5">DATA</td></tr>
</table>

Gambar 2.9 Format datagram IP

Pada gambar di atas diberikan format datagram IP. Setiap paket IP membawa data yang terdiri atas:

- *Version* berisi, versi dari protokol IP yang dipakai. Pada saat ini versi IP yang dipakai ialah IP versi 4.
- *Header Length*, berisi Panjang dari *header* paket IP ini dalam hitungan 32 bit word.
- *Type of Service*, berisi kualitas *service* yang dapat mempengaruhi cara penanganan paket IP ini.
- *Total length of Datagram*, panjang IP datagram total dalam ukuran byte
- *Identification, Flags,* dan *Fragment Offset*, berisi beberapa data yang berhubungan dengan fragmentasi paket. Paket yang dilewatkan melalui berbagai jenis jalur akan mengalami fragmentasi (dipecah pecah menjadi beberapa paket yang lebih kecil) sesuai dengan besar data maksimal yang bisa ditransmisikan melalui jalur tersebut
- *Time to Live*, berisi jumlah router/hop maksimal yang boleh dilewati paket IP. Setiap kali paket IP melewati satu router, isi dari field ini dikurangi satu. Jika TTL telah habis dan paket tetap belum sampai ke tujuan, paket ini akan dibuang dan router terakhir akan mengirimkan paket *ICMP time excedeed*. Hal ini dilakukan untuk mencegah paket IP terus menerus berada didalam network.
- *Protocol*, mengandung angka yang mengidentifikasikan protokol layer atas pengguna isi data dari paket IP ini.
- *Header Checksum*, berisi nilai *checksum* yang dihitung dari seluruh *field* dari *header* paket IP.

Sebelum dikirimkan, protokol IP terlebih dahulu menghitung checksum dari header paket IP tersebut untuk nantinya di hitung kembali di sisi penerima. Jika terjadi perbedaan, maka paket ini dianggap rusak dan dibuang.

- *IP address* pengirim dan penerima data, berisi alamat pengirim paket dan penerima paket

- Beberapa byte *option*, di antaranya:

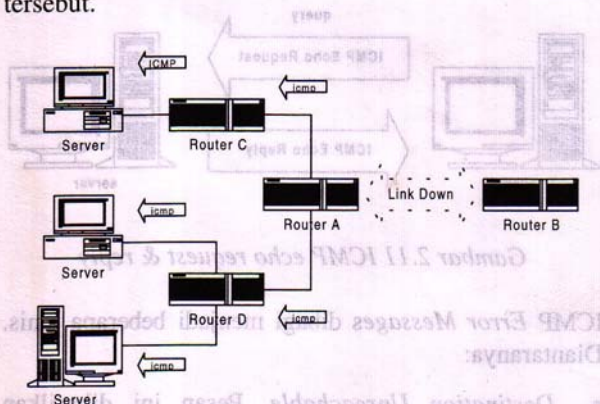
  Strict Source Route. Berisi daftar lengkap *IP address* dari *router* yang harus dilalui oleh paket ini dalam perjalanannya ke *host* tujuan. Selain itu paket balasan atas paket ini, yang mengalir dari *host* tujuan ke *host* pengirim, *diharuskan melalui router yang sama*.

  Loose Source Route. Dengan mengeset *option* ini, paket yang dikirim diharuskan singgah di beberapa *router* seperti yang disebutkan dalam *field option* ini. Jika diantara kedua *router* yang disebutkan terdapat *router* lain, paket masih diperbolehkan melalui *router* tersebut.

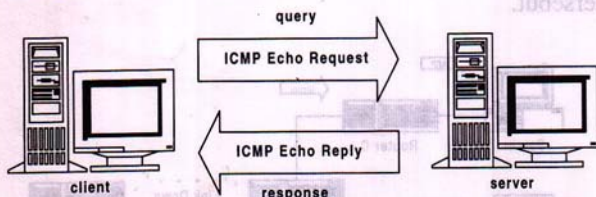
Bagaimana cara protokol IP melewatkan dan memilih jalur dalam mengirimkan paket? Hal ini akan dibahas sekilas di subbab berikutnya, yaitu routing sederhana, dan akan dibahas secara lengkap pada Bab 5.

ICMP (Internet Control Message Protocol)

ICMP (Internet Control Message Protocol) adalah protokol yang bertugas mengirimkan pesan-pesan kesalahan dan kondisi lain yang memerlukan perhatian khusus. Pesan/paket ICMP dikirim jika terjadi masalah pada layer IP dan layer atasnya (TCP/UDP).

Pada kondisi normal, Protokol IP berjalan baik dan menghasilkan proses penggunaan memori serta sumber daya transmisi yang efisien. Namun ada beberapa kondisi dimana koneksi IP terganggu, misalnya karena *router* yang *crash*, putusnya kabel, atau matinya host tujuan. Pada saat ini ICMP berperan membantu menstabilkan kondisi jaringan. Hal ini dilakukan dengan cara memberikan pesan pesan tertentu, sebagai respons atas kondisi tertentu yang terjadi pada jaringan tersebut.

Gambar 2.10 Timbulnya ICMP

Sebagai contoh, pada gambar di atas, hubungan antar *router* A dan B mengalami masalah, maka *Router* A akan secara otomatis mengirimkan paket *ICMP Destination Unreachable* ke *host* pengirim paket yang berusaha melewati *host* B menuju tujuannya. Dengan adanya pemberitahuan ini maka *host* tujuan tidak akan terus menerus berusaha mengirimkan paketnya melewati *router* B.

Contoh di atas hanya sebagian dari jenis pesan ICMP. Ada dua tipe pesan yang dapat dihasilkan oleh ICMP yaitu *ICMP Error Message* dan *ICMP Query Message*. ICMP *Error Message*, sesuai namanya, dihasilkan jika terjadi kesalahan pada jaringan. Sedangkan ICMP *Querry Message* ialah jenis pesan yang dihasilkan oleh protokol ICMP jika pengirim paket menginginkan informasi tertentu yang berkaitan dengan kondisi jaringan.

Gambar 2.11 ICMP echo request & reply

ICMP *Error Messages* dibagi menjadi beberapa jenis. Diantaranya:

- *Destination Unreachable*. Pesan ini dihasilkan oleh *router* jika pengiriman paket mengalami kegagalan akibat masalah putusnya jalur, baik secara fisik maupun secara logic. *Destination Unreachable* ini dibagi menjadi beberapa tipe. Beberapa tipe yang penting ialah:
- *Network Unreachable*, jika jaringan tujuan tak dapat dihubungi.
- *Host Unreachable*, jika *host* tujuan tak bisa dihubungi.

- *Protocol At Destination is Unreachable*, jika di tujuan tak tersedia protokol tersebut.
- *Port is Unreachable*, jika tidak ada port yang dimaksud pada tujuan.
- *Destination Network is Unknown*, jika *network* tujuan tak diketahui.
- *Destination Host is Unknown*, jika *Host* tujuan Tidak diketahui.
- *Time Exceeded.* Paket ICMP jenis ini dikirimkan jika isi field TTL dalam paket IP sudah habis dan paket belum juga sampai ke tujuannya. Sebagaimana telah diterangkan pada bagian IP di atas, tiap kali sebuah paket IP melewati satu router, nilai TTL dalam paket tersebut dikurangi satu. TTL ini diterapkan untuk mencegah timbulnya paket IP yang terus menerus berputar putar di network karena suatu kesalahan tertentu, sehingga menghabiskan sumberdaya jaringan yang ada.

  Field TTL ini pula yang digunakan oleh program *traceroute* untuk melacak jalannya paket dari satu host ke host lain. Program *traceroute* dapat melakukan pelacakan rute berjalannya IP dengan cara mengirimkan paket kecil UDP ke IP tujuan, dengan TTL yang di set membesar.

  Saat paket pertama dikirim, TTL di set satu, sehingga router pertama akan membuang paket ini dan mengirimkan paket *ICMP time exceeded.* Kemudian paket kedua dikirim, dengan TTL dinaikkan. Dengan naiknya TTL, paket ini sukses melewati router pertama namun dibuang oleh router kedua. Router ini pun mengirimkan *paket*

ICMP time exceeded. Dengan mendaftar nama-nama router yang mengirimkan paket *ICMP time exceeded* ini, akhirnya didapat seluruh nama router yang dilewati oleh paket UDP berikut:

```
> traceroute lyrics.ee.itb.ac.id

traceroute to gtw.EE.itb.ac.id (132.92.15.61), 30 hops max, 40 byte packets

1  indonesia-itb-ether.ITB.ac.id  (132.92.22.125)  1.120 ms  1.026 ms  0.987 ms

2  godam.ee.ITB.ac.id (132.92.22.112)  1.082 ms  3.835 ms  1.781 ms

3  167.205.7.10 (132.92.7.10) 6.476 ms 1.575 ms 3.682 ms

gtw.EE.ITB.ac.id (132.92.15.61) 2.370 ms 4.738 ms 2.054 ms
```

Gambar 2.12 Keluaran program traceroute

- *Parameter Problem.* Paket ini dikirim jika terdapat kesalahan parameter pada *header* Paket IP
- *Source Quench.* Paket ICMP ini dikirimkan jika *Router* atau tujuan mengalami kongesti. Sebagai respons atas paket ini, pihak pengirim paket harus memperlambat pengiriman paketnya
- *Redirect.* Paket ini dikirimkan jika router merasa *Host* mengirimkan paket IP melalui *router* yang salah. Paket ini seharusnya dikirimkan melalui *router* lain.

Sedangkan ICMP Query *Messages* terdiri atas:

- *Echo dan Echo Reply.* Bertujuan untuk memeriksa apakah sistem tujuan dalam keadaan aktif. Program ping merupakan program pengirim paket

ini. Responder harus mengembalikan data yang sama dengan data yang dikirimkan.

- *Timestamp dan Timestamp Reply*. Menghasilkan informasi waktu yang diperlukan sistem tujuan untuk memproses suatu paket
- *Address Mask*. Untuk mengetahui berapa netmask yang harus digunakan oleh suatu *host* dalam suatu *network*.

Sebagai paket pengatur kelancaran jaringan, paket ICMP tidak diperbolehkan membebani network. Karenanya, paket ICMP tidak boleh dikirim saat terjadi problem yang disebabkan oleh:

- Kegagalan pengiriman paket ICMP
- Kegagalan pengiriman paket broadcast atau multicast

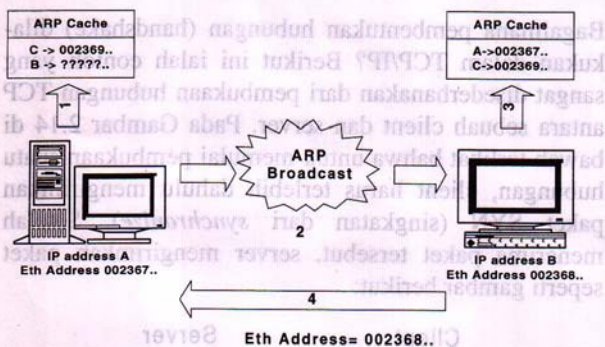
ARP (*Address Resolution Protocol*)

Dalam jaringan lokal, Paket IP biasanya dikirim melalui *card* ethernet. Untuk berkomunikasi mengenali dan berkomunikasi dengan ethernet lainnya, digunakan ethernet *Address*. Ethernet *address* ini besarnya 48 bit. Setiap *card* ethernet memiliki ethernet *address* yang berbeda-beda.

Pada saat hendak mengirimkan data ke komputer dengan IP tertentu, suatu *host* pada jaringan ethernet perlu mengetahui, di atas ethernet *address* yang manakah tempat IP tersebut terletak. Untuk keperluan pemetaan *IP address* dengan *ethernet Address* ini, digunakan protokol ARP (Address Resolution Protocol).

ARP bekerja dengan mengirimkan paket berisi *IP address* yang ingin diketahui alamat ethernetnya ke alamat *broadcast* ethernet. Karena dikirim ke alamat broadcast, semua card ethernet akan mendengar paket ini. *Host* yang merasa memiliki *IP address* ini akan membalas paket tersebut, dengan mengirimkan paket yang berisi pasangan *IP address* dan ethernet *Address*. Untuk menghindari seringnya permintaan jawaban seperti ini, jawaban ini disimpan di memori (ARP cache) untuk sementara waktu.

Cara kerja ARP dapat dituliskan sesuai algoritma berikut:

- Suatu host dengan IP address A ingin mengirim paket ke host dengan IP B pada jaringan lokal. Host pengirim memeriksa dulu ARP cache nya adakah hardware address untuk host dengan IP B
- Jika tidak ada, ARP akan mengirimkan paket ke alamat broadcast (sehingga seluruh jaringan mendengarnya). Paket ini berisi pertanyaan: "Siapa pemilik IP address B dan berapakah ethernet addressnya ?" Dalam paket ini juga disertakan IP address A dan ethernet addressnya.
- Setiap host di jaringan lokal menerima request tersebut dan memeriksa IP address masing-masing. Jika ia merasa paket tersebut bukan untuknya, dia tidak akan berusaha menjawab pertanyaan tersebut
- Host dengan IP address B yang mendengar request tersebut akan mengirimkan IP address & ethernet addressnya langsung ke host penanya.

Gambar 2.13 Cara kerja ARP

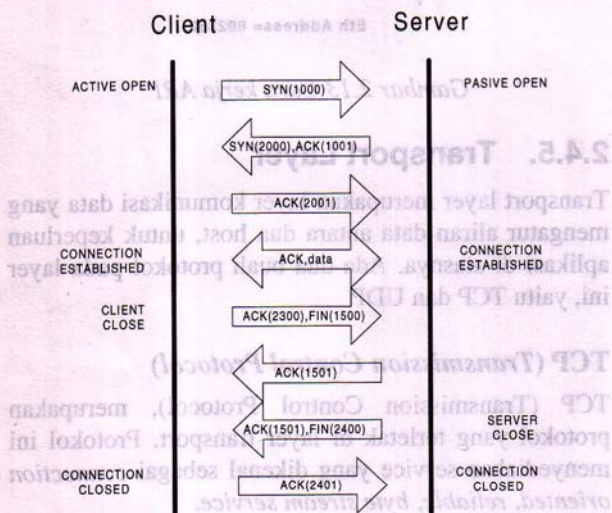
2.4.5. Transport Layer

Transport layer merupakan layer komunikasi data yang mengatur aliran data antara dua host, untuk keperluan aplikasi di atasnya. Ada dua buah protokol pada layer ini, yaitu TCP dan UDP

TCP (*Transmission Control Protocol*)

TCP (Transmission Control Protocol), merupakan protokol yang terletak di layer transport. Protokol ini menyediakan service yang dikenal sebagai *connection oriented, reliable, byte stream service.*

Apakah yang dimaksud dengan pernyataan di atas? connection oriented berarti sebelum melakukan pertukaran data, dua aplikasi pengguna TCP harus melakukan pembentukan hubungan (handshake) terlebih dulu. Reliable berarti TCP menerapkan proses deteksi kesalahan paket dan retransmisi. Byte Stream Service berarti paket dikirimkan dan sampai ke tujuan secara berurutan.

Bagaimana pembentukan hubungan (handshake) dilakukan dalam TCP/IP? Berikut ini ialah contoh yang sangat disederhanakan dari pembukaan hubungan TCP antara sebuah client dan server. Pada Gambar 2.14 di bawah terlihat bahwa untuk memulai pembukaan suatu hubungan, client harus terlebih dahulu mengirimkan paket SYN (singkatan dari *synchronize*). Setelah menerima paket tersebut, server mengirimkan paket seperti gambar berikut:

Gambar 2.14 Pembentukan dan pemutusan koneksi TCP

SYN miliknya serta acknowledgement (ACK) terhadap paket SYN sebelumnya. Saat client menerima paket ini, ia akan meng-ACKnowledge serta mengirimkan data miliknya. Pada saat ini terbentuklah koneksi TCP antara dua komputer, yaitu client dan server.

Angka dalam kurung yang mengikuti SYN pada gambar di atas adalah representasi dari sequence number. Sequence number ini pada awalnya dihasilkan secara acak. Setiap acknowledgement terhadap satu paket harus diikuti dengan sequence number yang lebih tinggi dibanding sequence number sebelumnya.

Untuk pemutusan hubungan TCP, kedua sisi harus mengirimkan paket yang berisi FIN (finish). Paket ini harus di-ACKnowledge oleh lawanya sebelum koneksi berakhir. Hal ini terlihat pada Gambar 2.14.

Apa sajakah yang dilakukan TCP agar reliabilitas pengiriman data terjamin? Untuk menjamin keandalan, TCP melakukan hal-hal berikut:

1. Data yang diterima oleh aplikasi dipecah menjadi segmen-segmen yang besarnya menurut TCP paling sesuai untuk mengirimkan data.
2. Ketika TCP menerima data dari mitranya, TCP mengirimkan acknowledgment (pemberitahuan bahwa ia telah menerima data)
3. Ketika TCP mengirimkan sebuah data, TCP mengaktifkan pewaktu (software timer) yang akan menunggu acknowledgement dari penerima segmen data tersebut. Jika sampai waktu yang ditentukan tidak diterima acknowledgement, data tersebut dikirimkan kembali oleh TCP.
4. Sebelum segmen data dikirim, TCP melakukan perhitungan checksum pada header dan data nya. Hal ini berbeda dengan protokol IP yang hanya melakukan perhitungan checksum pada headernya saja. Jika segmen yang diterima memiliki checksum yang tidak valid, TCP akan membuang

segmen ini dan berharap sisi pengirim akan melakukan retransmisi.

5. Karena segmen TCP dikirim menggunakan IP, dan datagram IP dapat sampai ke tujuan dalam keadaan tidak berurutan, segmen TCP yang dikirimnya pun dapat mengalami hal yang sama. Karenanya sisi penerima paket TCP harus mampu melakukan pengurutan kembali segmen TCP yang ia terima (resequencing), dan memberikan data dengan urutan yang benar ke aplikasi penggunanya.

6. Karena paket IP dapat terduplikasi di perjalanan, penerima TCP harus membuang data tersebut.

7. Untuk mencegah agar server yang cepat tidak membanjiri server yang lambat, TCP melakukan proses flow control. Setiap koneksi TCP memiliki buffer dengan ukuran yang terbatas. Sisi penerima TCP hanya memperbolehkan sisi pengirim mengirimkan data sebesar buffer yang ia miliki.

Bentuk segmen TCP terdapat pada gambar berikut.

<table>
<tr><td colspan="3">Source Port</td><td>Destination Port</td></tr>
<tr><td colspan="4">Sequence Number</td></tr>
<tr><td colspan="4">Acknowledgement Number</td></tr>
<tr><td>hdr</td><td>Resv</td><td>Control</td><td>Window</td></tr>
<tr><td colspan="3">Checksum</td><td>Urgent Pointer</td></tr>
<tr><td colspan="4">TCP Options</td></tr>
<tr><td colspan="4">Application Data</td></tr>
</table>

Gambar 2.15 Format segmen TCP

Segmen TCP terdiri atas beberapa field. *Source* dan *Destination* port adalah field berisi angka yang mengidentifikasi aplikasi pengirim dan penerima segmen TCP ini. *Sequence number* berisi nomor urut *byte stream* dalam data aplikasi yang dikirim. Setiap kali data ini sukses dikirim, pihak penerima data mengisi field acknowledgement number dengan sequence number berikutnya yang diharapkan penerima.

Header length berisi panjang header TCP. Dengan lebar 4 bit, field ini harus merepresentasikan panjang header TCP dalam satuan 4 byte. Jika 4 bit ini berisi 1 (1111 biner = 15 desimal), maka panjang header maksimal ialah 15 x 4 = 60 byte.

Field *window* pada gambar di atas diisi dengan panjang window (semacam buffer) penerimaan segment TCP, merupakan banyak byte maksimal yang bisa diterima tiap saat. Lebar field ini ialah 16 bit (2 byte). Sehingga nilai maksimalnya ialah 65535 .

UDP (*User Datagram Protocol*)

UDP (User Datagram Protocol) merupakan protokol transport yang sederhana. Berbeda dengan TCP yang connection oriented, UDP bersifat connectionless. Dalam UDP tidak ada *sequencing* (pengurutan kembali) paket yang datang, *acknowledgement* terhadap paket yang datang, atau *retransmisi* jika paket mengalami masalah di tengah jalan.

Kemiripan UDP dengan TCP ada pada penggunaan *port number*. Sebagaimana digunakan pada TCP, UDP menggunakan port number ini membedakan pengi-

riman datagram ke beberapa aplikasi berbeda yang terletak pada komputer yang sama.

Karena sifatnya yang *connectionless* dan *unreliable*, UDP digunakan oleh aplikasi aplikasi yang secara periodik melakukan aktivitas tertentu (misalnya query routing table pada jaringan lokal), serta hilangnya satu data akan dapat di atasi pada query periode berikutnya dan melakukan pengiriman data ke jaringan lokal. Pendeknya jarak tempuh datagram akan mengurangi resiko kerusakan data

Bersifat broadcasting atau multicasting. Pengiriman datagram ke banyak client sekaligus akan efisien jika prosesnya menggunakan metode *connectionless*.

Source Port	Destination Port
Datagram Length	Checksum
Application Data	

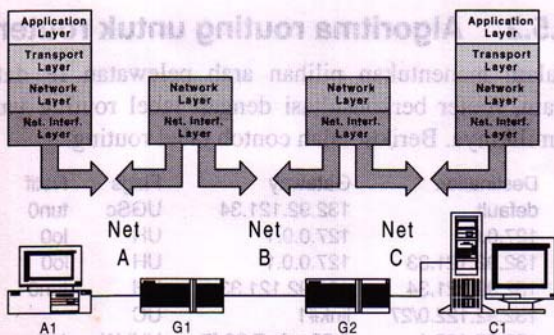
Gambar 2.16 Format datagram UDP

Pada Gambar 2.16 ditunjukkan format dari datagram UDP. *Source* dan *Destination* port memiliki fungsi yang sama seperti pada TCP. *Datagram length* berisi panjang datagram, sedangkan *checksum* berisi angka hasil perhitungan matematis yang digunakan untuk memeriksa kesalahan data.

2.5. Routing Sederhana

Routing berarti melewatkan paket IP menuju sasaran. Alat yang berfungsi melakukan routing paket ini

disebut sebagai *router*. Agar mampu melewatkan paket data antar jaringan, maka router minimal harus memiliki dua buah network interface.

Proses routing dilakukan secara *hop by hop*. IP tidak mengetahui jalur keseluruhan menuju tujuan setiap paket. IP routing hanya menyediakan IP address dari router berikutnya (*next hop router*) yang menurutnya "lebih dekat" ke host tujuan. Bagaimana contohnya ?

Gambar 2.17 Routing paket

Sistem hanya bisa mengirim paket pada divais lain yang terhubung kedalam satu jaringan fisik yang sama. Paket dari A1 dengan tujuan C1 diforward melalui router G1 dan G2. Host A1 pertama kali mengirim paket ke router G1 (karena G1 terhubung ke tempat di mana host A1 berada). Kemudian router G1 mengirimkan paket ke router G2 melalui network B. Dan akhirnya G2 yang juga terhubung ke network C langsung menyampaikan paket ke address tujuan, host C1

2.5.1. Algoritma routing untuk host

Proses routing yang dilakukan oleh host cukup sederhana. Jika host tujuan terletak di jaringan yang sama atau terhubung langsung, IP datagram dikirim langsung ke tujuan. Jika tidak, IP datagram dikirim ke default router. Router ini yang akan mengatur pengiriman IP selanjutnya, hingga sampai ke tujuannya.

2.5.2. Algoritma routing untuk router

Dalam menentukan pilihan arah pelewatan IP datagram, router berkonsultasi dengan tabel routing yang dimilikinya. Berikut ialah contoh tabel routing.

Destination	Gateway	Flags	Netif
default	132.92.121.34	UGSc	tun0
127.0.0.1	127.0.0.1	UH	lo0
132.92.121.33	127.0.0.1	UH	lo0
132.92.121.34	132.92.121.33	UH	tun0
132.92.122.0/27	link#1	UC	
132.92.122.1	0:80:ad:a7:96:f5	UHLW	lo0
132.92.122.31	ff:ff:ff:ff:ff:ff	UHLWb	ed0
132.92.122.32/27	132.92.122.3	UGSc	ed0

Gambar 2.18 Tabel routing

Dari gambar di atas terlihat tabel routing ini berisi:

- IP address tujuan
- IP address next hop router (gateway)
- Flag. Flag ini menyatakan jenis routing. Jenis routing ini akan diterangkan secara khusus pada bab IP routing
- Spesifikasi Network interface tempat datagram dilewatkan.

Dalam proses meneruskan paket ke tujuan, IP router melakukan hal-hal berikut:

1. Mencari di tabel routing , entry yang cocok dengan IP address tujuan. Jika ditemukan, paket akan di kirim ke next hop router atau interface yang terhubung langsung dengannya (*directly connected interface*)
2. Mencari di tabel routing, entry yang cocok dengan alamat network dari network tujuan. Jika ditemukan, paket dikirim ke next hop router tersebut.
3. Mencari di tabel routing, entry yang bertanda default. Jika ditemukan, paket dikirim ke router tersebut.

Tabel routing ini dihasilkan oleh program protokol routing. Konfigurasi program protokol routing ini akan diterangkan secara jelas pada Bab 5.

2.6. DNS Domain dan Mapping

Setiap network interface yang terhubung ke jaringan TCP/IP memiliki IP address yang unik. IP address yang 32 bit ini bukan merupakan hal yang mudah diingat. Nama host (hostname) biasanya digunakan untuk mengingat suatu komputer.

Program komputer sendiri sebenarnya tidak memerlukan hostname. Setiap host di internet tetap menggunakan IP address untuk berhubungan dengan host lain. Hostname justru diperlukan oleh manusia untuk mempermudah tugasnya mengoperasikan jaringan komputer.

Sebagai contoh, coba ketikkan kedua URL berikut pada browser Anda.

http://www.yahoo.com

http:// 204.71.200.72

Kedua URL ini akan menghasilkan halaman web yang sama, yaitu http://www.yahoo.com. Perbedaanya hanyalah saat Anda mengetikkan http://www.yahoo.com, komputer Anda terlebih dahulu mencari IP address dari komputer host dengan nama http://www.yahoo.com.

2.6.1. Metode Memetakan Hostname ke IP address

Terdapat dua metode yang digunakan untuk mendapatkan IP address dari suatu hostname. Metode pertama ialah menggunakan host table. Host table merupakan file yang berisi kombinasi antara nama host dengan IP address host tersebut.

Untuk jaringan kecil, penggunaan host table masih dimungkinkan. Namun ketika jaringan menjadi sangat besar. Penggunaan host table menjadi tidak efisien (karena semua nama host dan IP address harus masuk dalam host table).

Untuk menanggulangi kelemahan sistem host table ini dibuatlah DNS (*Domain Name Service*). DNS merupakan sistem database terdistribusi yang tidak banyak dipengaruhi oleh bertambahnya database. DNS menjamin informasi host terbaru akan disebarkan ke jaringan bila diperlukan. Jika server DNS menerima permintaan informasi tentang host yang tidak dia

ketahui, ia akan bertanya pada *authoritative DNS server* (sembarang server yang bertanggung jawab untuk memberikan informasi akurat tentang domain yang diminta). Ketika authoritative server memberikan jawabannya, server lokal menyimpan jawabannya untuk penggunaan mendatang. Jadi, apabila setelah itu ada permintaan informasi yang sama ia langsung menjawabnya.

2.7. Ringkasan

Arsitektur dasar TCP/IP dibentuk secara modular untuk menangani bermacam masalah komunikasi data. Modularitas ini memungkinkan TCP/IP secara mudah diimplementasikan pada berbagai macam jenis komputer dan peralatan jaringan komputer. Secara fisik, jaringan komputer dibentuk dengan menambahkan peralatan network interface dan netwroking device (router, bridge, repeater) lainnya. Secara software TCP/IP terdiri atas berbagai modul software yang memiliki berbagai fungsi. IP dan routing berfungsi mengatur sampainya paket ke tujuan yang tepat, sedangkan TCP dan UDP berfungsi mengatur komunikasi antara dua atau lebih aplikasi. TCP lebih reliable dibanding UDP karena TCP menerapkan teknik sequencing dan retransmission.

Bab 3
IP Address

Pada bab sebelumnya, kita mempelajari arsitektur TCP/IP yang dilihat sebagai lapisan-lapisan komponen penyusunnya. Identitas sebuah host sering digolongkan ke dalam *nama, alamat,* dan *rute* host tersebut. Nama menunjukkan benda *apakah* itu, alamat menunjukkan *tempat* dia berada, dan rute memberitahu cara mencapainya.

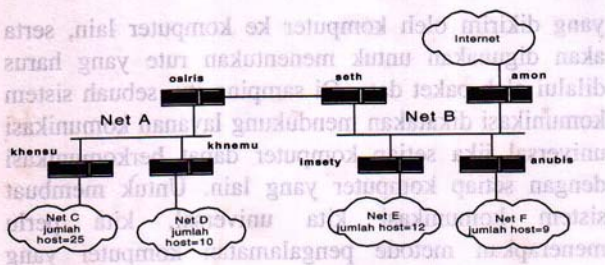
Bab ini akan menjelaskan kepada Anda, sebuah resep penting yang akan menyembunyikan detail-detail arsitektur TCP/IP tersebut sehingga internet tampil sebagai sesuatu yang seragam dan tunggal. Resep tersebut ialah *IP address*. *Rute* dan *pemetaan alamat* ke dalam *nama* yang mudah diingat akan dibahas pada bab selanjutnya.

3.1. Pendahuluan

Dalam mendesain sebuah jaringan komputer yang terhubung ke internet, kita perlu menentukan IP address untuk tiap komputer dalam jaringan tersebut. Penentuan IP address ini termasuk bagian terpenting dalam pengambilan keputusan desain. Hal ini disebabkan oleh IP address (yang terdiri atas bilangan 32-bit ini) akan ditempatkan dalam *header* setiap paket data

yang dikirim oleh komputer ke komputer lain, serta akan digunakan untuk menentukan rute yang harus dilalui oleh paket data. Di samping itu, sebuah sistem komunikasi dikatakan mendukung layanan komunikasi universal jika setiap komputer dapat berkomunikasi dengan setiap komputer yang lain. Untuk membuat sistem komunikasi kita universal, kita perlu menerapkan metode pengalamatan komputer yang telah diterima di seluruh dunia.

Dengan menentukan IP address, kita melakukan pemberian identitas yang universal bagi setiap interface komputer. Setiap komputer yang tersambung ke internet setidaknya harus memiliki sebuah IP address pada setiap interfacenya. Dalam penerapan sehari-hari, kita dapat melihat sebuah komputer memiliki lebih dari satu interface, misal ada sebuah card Ethernet dan sebuah interface serial (Gambar 3.1). Maka, kita harus memberi dua IP address kepada komputer tersebut masing-masing untuk setiap interfacenya. Jadi, sebuah IP address sesungguhnya tidak merujuk ke sebuah komputer, tetapi ke sebuah interface.

Konsep dasar pengalamatan di internet ialah awalan (*prefix*) pada IP address dapat digunakan sebagai dasar pengambilan keputusan dalam pemilihan rute paket data ke alamat tujuan. Misalnya, 16 bit pertama menandakan jaringan PT Jaya, 20 bit pertama menandakan jaringan pada kantor Administrasi perusahaan yang sama, 26 bit pertama menandakan segmen jaringan Ethernet pada kantor tersebut, dan keseluruhan 32 bit menandakan interface komputer tertentu pada jaringan Ethernet tersebut.

Gambar 3.1 Sebuah komputer dengan dua interface dan dua buah IP address

Dengan demikian, kesalahan dalam mendesain dapat menyebabkan sebuah komputer dapat dicapai oleh sebuah IP address, tetapi tidak dapat dicapai oleh IP address yang lain. Jalan keluar yang paling sederhana adalah dengan memilih interface yang paling bagus dan mengumumkan IP addressnya sebagai IP address primer komputer tersebut.

3.2. Format IP address

3.2.1. Bentuk biner

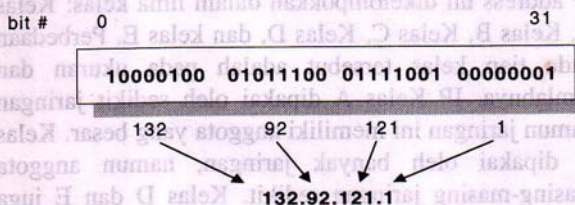
IP address merupakan bilangan biner 32 bit yang dipisahkan oleh tanda pemisah berupa tanda titik setiap 8 bitnya. Tiap 8 bit ini disebut sebagai oktet. Bentuk IP address adalah sebagai berikut

xxxxxxxx . xxxxxxxx . xxxxxxxx . xxxxxxxx

Setiap simbol "x" dapat digantikan oleh angka 0 dan 1, misalnya sebagai berikut:

10000100 . 1011100 . 1111001 . 00000001

3.2.2. Bentuk dotted decimal

Notasi IP address dengan bilangan biner seperti di atas tidaklah mudah dibaca. Untuk membuatnya lebih mudah dibaca dan ditulis, IP address sering ditulis sebagai 4 bilangan desimal yang masing-masing dipisahkan oleh sebuah titik. Format penulisan seperti ini disebut "dotted-decimal notation" (notasi desimal bertitik). Setiap bilangan desimal tersebut merupakan nilai dari satu oktet (delapan bit) IP address. Gambar 3.3 berikut memperlihatkan bagaimana sebuah IP address yang ditulis dengan notasi dotted-decimal:

Gambar 3.2 Notasi Dotted-Decimal

3.3. Kelas IP Address dan Artinya

Jika dilihat dari bentuknya, IP address terdiri atas 4 buah bilangan biner 8 bit. Nilai terbesar dari bilangan biner 8 bit ialah 255 (= $2^7 + 2^6 + 2^5 + 2^4 + 2^3 + 2^2 + 2 + 1$). Karena IP address terdiri atas 4 buah bilangan 8 bit, maka jumlah IP address yang tersedia ialah 255 x 255 x 255 x255. IP address sebanyak ini harus dibagi bagikan ke seluruh pengguna jaringan internet di seluruh dunia.

Untuk mempermudah proses pembagiannya, IP address dikelompokkan dalam kelas-kelas. Dasar pertimbangan pembagian IP address ke dalam kelas-kelas adalah untuk memudahkan pendistibusian pendaftaran IP address. Dengan memberikan sebuah *ruang nomor jaringan* (beberapa blok IP address) kepada ISP (*Internet Service Provider*) di suatu area diasumsikan penanganan komunitas lokal tersebut akan lebih baik, dibandingkan dengan jika setiap pemakai individual harus meminta IP address ke otoritas pusat, yaitu *Internet Assigned Numbers Authority* (IANA).

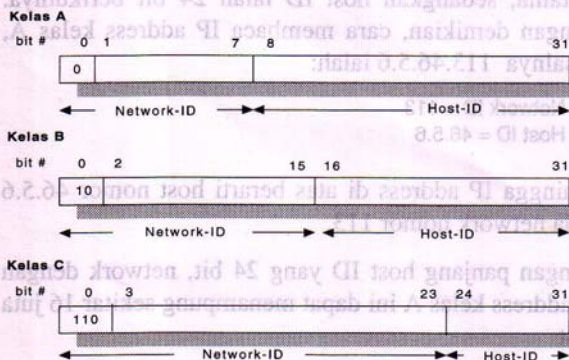
IP address ini dikelompokkan dalam lima kelas: Kelas A, Kelas B, Kelas C, Kelas D, dan kelas E. Perbedaan pada tiap kelas tersebut adalah pada ukuran dan jumlahnya. IP Kelas A dipakai oleh sedikit jaringan namun jaringan ini memiliki anggota yang besar. Kelas C dipakai oleh banyak jaringan, namun anggota masing-masing jaringan sedikit. Kelas D dan E juga didefinisikan, tetapi tidak digunakan dalam penggunaan normal. Kelas D diperuntukkan bagi jaringan multicast, dan Kelas E untuk keperluan eksperimental.

3.3.1. Network ID dan host ID

Pembagian kelas-kelas IP address didasarkan pada dua hal: *network ID* dan *host ID* dari suatu IP address.

Setiap IP address selalu merupakan sebuah pasangan dari *network-ID* (identitas jaringan) dan *host-ID* (identitas host dalam jaringan tersebut). *Network-ID* ialah bagian dari IP address yang digunakan untuk menunjukkan jaringan tempat komputer ini berada. Sedangkan *host-ID* ialah bagian dari IP address yang digunakan untuk menunjukkan workstation, server,

router, dan semua host TCP/IP lainnya dalam jaringan tersebut. Dalam satu jaringan, *host-ID* ini harus unik (tidak boleh ada yang sama).

Sedangkan dari sisi praktisnya, setiap IP address harus memiliki salah satu bentuk dari ketiga bentuk pertama pada Gambar 3.2.

Gambar 3.3 Prinsip format IP address berdasarkan kelas

3.3.2. Kelas A

Karakteristik:

Format	: 0nnnnnnn hhhhhhhh hhhhhhhh hhhhhhhh
Bit pertama	: 0
Panjang NetID	: 8 bit
Panjang HostID	: 24 bit
Byte pertama	: 0 -127
Jumlah	: 126 Kelas A (0 dan 127 dicadangkan)
Range IP	: 1.xxx.xxx.xxx sampai 126.xxx.xxx.xxx
Jumlah IP	: 16.777.214 IP address pada tiap Kelas A

IP address kelas A diberikan untuk jaringan dengan jumlah host yang sangat besar. Bit pertama dari IP address kelas A selalu di set 0 (nol) sehingga byte terdepan dari IP address kelas A selalu bernilai antara angka 0 dan 127.

Pada IP address kelas A, network ID ialah delapan bit pertama, sedangkan host ID ialah 24 bit berikutnya. Dengan demikian, cara membaca IP address kelas A, misalnya 113.46.5.6 ialah:

Network ID = 113
Host ID = 46.5.6

Sehingga IP address di atas berarti host nomor 46.5.6 pada network nomor 113.

Dengan panjang host ID yang 24 bit, network dengan IP address kelas A ini dapat menampung sekitar 16 juta host

3.3.3. Kelas B

Karakteristik:

Format	: 10nnnnnn nnnnnnnn hhhhhhhh hhhhhhhh
2 bit pertama	: 10
Panjang NetID	: 16 bit
Panjang HostID	: 16 bit
Byte pertama	: 128-191
Jumlah	: 16.384 Kelas B
Range IP	: 128.0.xxx.xxx sampai 191.155.xxx.xxx
Jumlah IP	: 65.532 IP address pada tiap Kelas B

IP address kelas B biasanya dialokasikan untuk jaringan berukuran sedang dan besar. Dua bit pertama dari IP address kelas B selalu di set 10 (satu nol)

sehingga byte terdepan dari IP address kelas B selalu bernilai antara 128 hingga 191.

Pada IP address kelas B, network ID ialah enam belas bit pertama, sedangkan host ID ialah 16 bit berikutnya. Dengan demikian, cara membaca IP address kelas A, misalnya 132.92.121.1 ialah:

Network ID = 132.92
Host ID = 121.1

Sehingga IP address di atas berarti host nomor 121.1 pada network nomor 132.92

Dengan panjang host ID yang 16 bit, network dengan IP address kelas B ini dapat menampung sekitar 65000 host

3.3.4. Kelas C

Karakteristik:

Format	: 110nnnnn nnnnnnnn nnnnnnnn hhhhhhhh
3 bit pertama	: 110
Panjang NetID	: 24 bit
Panjang HostID	: 8 bit
Byte pertama	: 192-223
Kelas	: 2.097.152 Kelas C
Range IP	: 192.0.0.xxx sampai 223.255.255.xxx
Jumlah IP	: 254 IP address pada tiap Kelas C

IP address kelas C awalnya digunakan untuk jaringan berukuran kecil (misalnya LAN). Tiga bit pertama dari IP address kelas C selalu berisi 111. Bersama 21 bit berikutnya, angka ini membentuk network ID 24 bit. Host-ID ialah 8 bit terakhir. Dengan konfigurasi ini, bisa dibentuk sekitar dua juta network dengan masing-masing network memiliki 256 IP address.

3.3.5. Kelas D

Karakteristik:

Format	: 1110mmmm mmmmmmmm mmmmmmmm mmmmmmmm
4 bit pertama	: 1110
Bit multicast	: 28 bit
Byte inisial	: 224-247
Deskripsi	: Kelas D adalah ruang alamat multicast (RFC 1112)

IP address kelas D digunakan untuk keperluan IP multicasting. 4 bit pertama IP address kelas D di set 1110. Bit-bit berikutnya diatur sesuai keperluan multicast group yang menggunakan IP address ini. Dalam multicasting tidak dikenal network bit dan host bit.

3.3.6. Kelas E

Karakteristik:

Format	: 1111rrrr rrrrrrrr rrrrrrrr rrrrrrrr
4 bit pertama	: 1111
Bit cadangan	: 28 bit
Byte inisial	: 248-255
Deskripsi	: Kelas E adalah ruang alamat yang dicadangkan untuk keperluan ekperimental

IP address kelas E tidak digunakan untuk umum. 4 bit pertama IP address ini di set 1111.

Selain network ID, istilah lain yang digunakan untuk menyebut bagian IP address yang menunjukkan jaringan ialah *Network Prefix*. Biasanya dalam menuliskan network prefix suatu kelas IP address digunakan tanda garis miring (*slash*) "/", yang diikuti dengan

angka yang menunjukkan panjang network prefix ini dalam bit.

Misalnya, ketika menuliskan network kelas A dengan alokasi IP 12.xxx.xxx.xxx, network prefixnya dituliskan sebagai: 12/8. Angka delapan menunjukkan jumlah bit yang digunakan oleh network prefix.

Untuk menunjuk satu network kelas B 167.205.xxx.xxx, digunakan: 167.205/16. Angka 16 merupakan panjang bit untuk network prefix pada IP address kelas B.

3.4. Pengalokasian IP Address

Pengalokasian IP address pada dasarnya ialah proses memilih network ID dan host ID yang tepat untuk suatu jaringan. Tepat atau tidaknya konfigurasi ini tergantung dari tujuan yang hendak dicapai, yaitu mengalokasikan IP address seefisien mungkin.

3.4.1. Aturan dasar pemilihan network ID dan host ID

Terdapat beberapa aturan dasar dalam menentukan network ID dan host ID yang hendak digunakan. Aturan tersebut ialah

Network ID tidak boleh sama dengan 127

Network ID 127 tidak dapat digunakan karena ia secara default digunakan untuk keperluan loopback. Loopback ialah IP address yang digunakan komputer untuk menunjuk dirinya sendiri.

Network ID dan host ID tidak boleh sama dengan 255 (seluruh bit di set 1)

Seluruh bit dari network ID dan host ID tidak boleh semuanya di set 1. Jika hal ini dilakukan, network ID atau host ID tersebut akan diartikan sebagai alamat broadcast. ID broadcast merupakan alamat yang mewakili seluruh anggota jaringan. Pengiriman paket ke alamat broadcast akan menyebabkan paket ini didengarkan oleh seluruh anggota network tersebut.

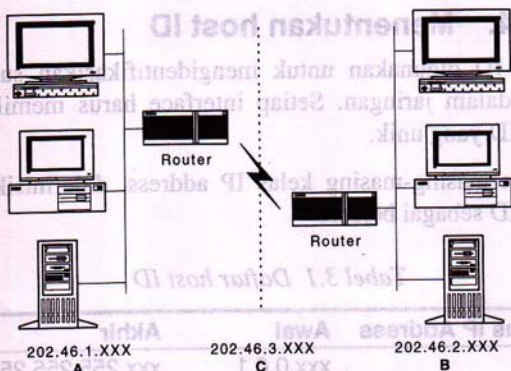
Network ID dan host ID tidak boleh 0 (nol)

Network ID dan host ID tidak boleh semua bitnya 0 (nol). IP address dengan host ID 0 diartikan sebagai alamat network. Alamat network ialah alamat yang digunakan untuk menunjuk suatu jaringan, dan tidak menunjukkan suatu host

Host ID harus unik dalam satu network

Dalam satu network, tidak boleh ada dua host yang memiliki host ID yang sama.

3.4.2. Contoh 1: Pengalokasian IP Address

Asumsikan kita diberi hak mengelola 3 IP address kelas C 202.46.1.xxx, 202.46.2.xxx dan 202.46.3.xxx. Sedangkan jaringan yang kita miliki ialah sebagai berikut:

Gambar 3.4 Dua network terhubung melalui router via WAN

3.4.3. Menentukan netwok ID

Network ID digunakan untuk menunjukkan host TCP/IP yang terletak pada network yang sama. Semua host pada satu jaringan harus memiliki network ID yang sama. Jika antara network dihubungkan oleh router, network ID tambahan dibutuhkan untuk hubungan antar router tersebut

Pada gambar 3.4, terdapat tiga jaringan, yaitu jaringan A, B dan C. Jaringan C merupakan penghubung antar-jaringan A dan B. Masing-masing jaringan ini diberi network ID 202.46.1.xxx, 202.46.2.xxx, dan 202.46.3.xxx

3.4.4. Menentukan host ID

Host ID digunakan untuk mengidentifikasikan suatu host dalam jaringan. Setiap interface harus memiliki host ID yang unik.

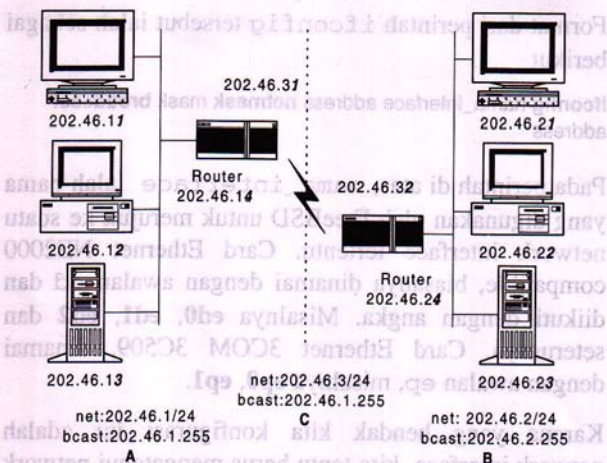
Untuk masing-masing kelas IP address, didefinisikan host ID sebagai berikut

Tabel 3.1 Daftar host ID

Kelas IP Address	Awal	Akhir
A	xxx.0.0.1	xxx.255.255.254
B	xxx.xxx.0.1	xxx.xxx.255.254
C	xxx.xxx.xxx.1	xxx.xxx.xxx.254

Tabel 3.1 menunjukkan host ID awal untuk IP address kelas A adalah 0.0.1, dan bukan 0.0.0. Host ID 0.0.0 ini digunakan untuk keperluan alamat network. Sebagai contoh, IP address 12.0.0.0 tidaklah menunjukkan host 0.0.0 pada jaringan 12, namun menunjukkan network 12/8 itu sendiri. Dengan kata lain, IP 12.0.0.0 digunakan sebagai alamat network.

Pada tabel di atas juga ditunjukkan bahwa host ID terakhir pada suatu network kelas C ialah 254. Host ID 255 digunakan sebagai alamat broadcast. Jika suatu paket IP dikirimkan ke alamat ini, seluruh host dalam satu jaringan akan mendengarkan paket tersebut.

Berdasarkan daftar di atas pula, untuk kelas C, host ID yang boleh dialokasikan adalah 1 hingga 254. Oleh karenanya masing-masing anggota jaringan kelas C pada Gambar 3.4 di atas diharuskan memilih salah satu dari 254 host ID di atas. Hasilnya terlihat pada Gambar 3.5.

Gambar 3.5 Host ID jaringan kelas C

3.4.5. Konfigurasi network Interface pada FreeBSD

Setelah, IP address, netmask dan broadcast address didefinisikan, ketiganya harus di set di komputer yang bersangkutan. Pada sistem operasi FreeBSD, terdapat perintah yang biasa digunakan untuk mengkonfigurasi interface. Perintah tersebut ialah **`ifconfig`**.

Perintah `ifconfig` digunakan untuk mengecek atau mengkonfigurasi sebuah network interface. Yang dapat dikonfigurasi oleh perintah ini antara lain

- IP address
- Subnet Mask
- Broadcast address

Format dari perintah `ifconfig` tersebut ialah sebagai berikut

ifconfig nama_interface address **netmask** mask **broadcast** address

Pada perintah di atas, `nama_interface` ialah nama yang digunakan oleh FreeBSD untuk merujuk ke suatu network interface tertentu. Card Ethernet NE2000 compatible, biasanya dinamai dengan awalan `ed` dan diikuti dengan angka. Misalnya **ed0**, **ed1**, **ed2** dan seterusnya. Card Ethernet 3COM 3C509, dinamai dengan awalan `ep`, misalnya **ep0**, **ep1**.

Karena yang hendak kita konfigurasi ini adalah network interface, kita tentu harus mengetahui network interface apa saja yang tersedia pada host kita ini. Biasanya pada saat booting, seluruh interface yang ada pada sistem kita nampak di console. hal ini dapat juga dilihat dengan menggunakan perintah **`dmesg`**.

Mendefinisikan interface

Sebagai contoh, perintah ifconfig berikut ini digunakan untuk mengkonfigurasi interface "ed1" pada suatu host dengan IP address 202.46.2.4 dan netmask 255.255.255.0

ifconfig ed1 202.46.2.4 **netmask** 255.255.255.0 **broadcast** 202.46.2.255

setelah dikonfigurasi, kita dapat melihat interface kita ini. Untuk itu digunakan perintah

ifconfig nama_interface

contoh:

```
% ifconfig ed1

ed0:
flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAS
T>

mtu 1500 inet 202.46.2.4 netmask 0xffffff00 broadcast
202.46.2.255

ether 00:80:ad:a7:96:f5
```

Berdasarkan hasil perintah ini, terlihat bahwa interface `ed1` sedang operasional (Up), terhubung ke jaringan yang mendukung mode broadcast (dalam hal ini ethernet), simplex (interface hanya bisa digunakan satu arah tiap saat), serta mendukung pengalamatan multicast.

Menyalakan dan mematikan interface

`ifconfig` memiliki dua argument, yaitu "up" dan "down" yang berguna untuk menandai sebuah interface. Argument "up" digunakan untuk menandai interface tersebut sedang operasional, sehingga ia siap untuk digunakan. Argument "down" sebaliknya, menandai network interface sedang dalam keadaan tidak operasional sehingga ia tak dapat digunakan.

Perintah "down" ini dipakai jika kita ingin melakukan konfigurasi ulang terhadap sebuah interface. Beberapa parameter konfigurasi, seperti IP Address misalnya, tak dapat diubah kecuali jika interfacenya sedang dalam keadaan "down" sehingga untuk melakukan perubahan IP Address, mula-mula interface tersebut di "down" dahulu, dilakukan rekonfigurasi, baru kemudian ia di "up" kembali. Contoh

```
% ifconfig ed0 down
% ifconfig ed0 167.205.9.1 up
```

Setelah perintah ini, interface ed0 beroperasi dengan IP Address yang baru.

3.4.6. Subnetting

Kegunaan subnetting

Setiap organisasi yang terhubung ke Internet memperoleh sebuah network ID dari internic (**http://www.internic.net**). Network ID ini memiliki ukuran bermacam macam, mulai dari kelas A, B, hingga kelas C.

Network ID dengan ukuran tertentu ini jarang sekali langsung digunakan untuk membentuk satu jaringan. Biasanya sebuah organisasi memiliki lebih dari satu jaringan/LAN, yang masing-masing jumlah hostnya tidak sebesar jumlah maksimal host yang disediakan oleh satu kelas IP address.

Ada beberapa alasan yang menyebabkan sebuah organisasi memerlukan lebih dari satu LAN agar dapat mencakup seluruh organisasi, yaitu:

Teknologi yang berbeda: khususnya dalam sebuah lingkungan riset, yang terdapat beberapa LAN karena terdapat peralatan yang harus didukung oleh Ethernet, dan yang lain oleh jaringan token-ring.

Keterbatasan teknologi: sebagian besar teknologi LAN memiliki batas kemampuan berdasarkan pada parameter elektrikal, jumlah host yang terhubung, dan panjang total dari kabel. Batas ini paling sering dicapai oleh faktor panjang kabel.

Kongesti pada jaringan: Sebuah LAN dengan 254 Host misalnya akan memiliki performansi yang kurang baik, dibandingkan dengan LAN berukuran kecil, jika teknologi yang digunakan ialah ethernet. Sekian banyak host yang menggunakan satu media bersama-sama untuk berbicara satu dengan lainnya akan membuat kesempatan akses masing-masing host terhadap jaringan menjadi kecil. Selain itu dalam sebuah LAN mungkin terdapat beberapa host yang memonopoli penggunaan bandwidth. Jalan keluar yang paling umum adalah memisahkannya kedalam sebuah kelompok kecil dan menempatkannya pada kabel yang terpisah.

Hubungan point-to-point: karena jauhnya dua lokasi sebuah kampus, maka diperlukan teknologi LAN tertentu yang dapat mencakup "local area" ini. Biasanya digunakanlah hubungan point-to-point berkecepatan tinggi untuk menghubungkan beberapa LAN tersebut.

Karena alasan alasan di atas, network ID yang dimiliki oleh suatu organisasi dipecah lagi menjadi be-berapa network ID lain dengan jumlah anggota jaring-an yang lebih kecil. Teknik ini dinamakan *subnetting* dan jaringannya dinamakan *subnet* (subnetwork)

Subnet mask

Subnet mask ialah angka biner 32 bit yang digunakan untuk:

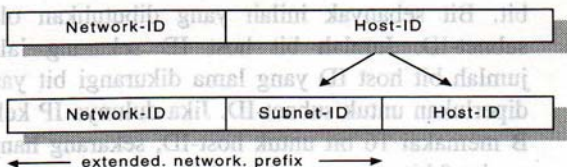
- Membedakan network ID dan host ID.
- Menunjukkan letak suatu host, apakah berada di jaringan lokal atau jaringan luar.

Tabel 3.2 Subnet mask untuk tiap kelas IP address

Kelas IP address	Bit subnet mask	Subnet dlm dotted decimal
Kelas A	11111111 . 00000000 . 00000000 . 00000000	255.0.0.0
Kelas B	11111111 . 11111111 . 00000000 . 00000000	255.255.0.0
Kelas C	11111111 . 11111111 . 11111111 . 00000000	255.255.255.0

Pada subnet mask, seluruh bit yang berhubungan dengan network ID di set 1. Sedangkan bit yang berhubungan dengan host-ID di set 0. IP address kelas A misalnya, secara default memiliki subnet mask 255.0.0.0 yang menunjukkan batas antara network-ID dan host-ID IP address kelas A

Subnet mask juga digunakan untuk menentukan letak suatu host, apakah di jaringan lokal, atau di jaringan luar. Hal ini diperlukan untuk operasi pengiriman paket IP. Dengan melakukan operasi AND antara subnet mask dengan IP address asal dan IP address tujuan, serta membandingkan hasilnya, dapat diketahui arah tujuan paket IP tersebut. Jika kedua hasil operasi tersebut sama, maka host tujuan terletak di jaringan lokal, dan paket IP dikirim langsung ke host tujuan. Jika hasilnya berbeda, host tujuan terletak diluar jaringan lokal, sehingga paket pun dikirim ke default router.

Dalam subnetting, proses yang dilakukan ialah memakai sebagian bit host ID untuk membentuk subnet-ID. Dengan demikian jumlah bit yang digunakan untuk Host-ID menjadi lebih sedikit. Semakin panjang subnet ID, jumlah subnet yang dapat dibentuk semakin banyak, namun jumlah host dalam tiap subnet menjadi semakin sedikit. Hal ini ditunjukkan pada Gambar 3.6

Gambar 3.6 Subnet-ID

Dengan adanya subnet-ID ini, network prefix tidak lagi sama dengan network ID. Network prefix yang baru ialah network ID ditambah subnet-ID. Untuk membedakannya dengan network prefix lama, digunakan istilah *extended network prefix*.

Sebagai contoh, IP address kelas B 132.92.121.1 secara default memiliki subnet mask 255.255.0.0. Dengan subnet mask ini, IP address di atas berarti host nomor 121.1 pada network 132.92/16.

Jika alokasi kelas B 132.92.xxx.xxx ini ingin dibagi-bagi untuk digunakan pada jaringan jaringan kecil, subnet mask yang digunakan harus diubah.

Misalnya kita ingin membagi alokasi kelas B di atas menjadi jaringan kecil kelas C (254 host ID). Cara menentukan subnet masknya ialah

- Mengubah jumlah network yang dibutuhkan menjadi bilangan biner. Satu network kelas B dapat diubah menjadi 255 network kelas C. Angka 255 jika direpresentasikan dalam biner ialah 11111111.
- Menghitung jumlah bit yang dibutuhkan untuk merepresentasikan angka tersebut. Untuk merepresentasikan angka 255 dalam biner dibutuhkan 8

bit. Bit sebanyak inilah yang dibutuhkan oleh subnet-ID. Jumlah bit host ID sekarang ialah jumlah bit host ID yang lama dikurangi bit yang diperlukan untuk subnet-ID. Jika dulunya IP kelas B memakai 16 bit untuk host-ID, sekarang hanya tersisa 8 bit saja.

- Mengisi Subnet-ID ini dengan bit 1. Sehingga subnet mask yang baru yaitu:

11111111 . 11111111 . 11111111 . 00000000

132.92.121.1	10000100	01011100	01111001	00000011	
255.255.255.0	11111111	11111111	11111111	00000000	AND
132.92.121.0	10000100	01011100	01111001	00000000	
	Net ID		Subnet ID	Host ID	

Dengan adanya subnet mask baru ini, IP address 132.92.121.1 dibaca sebagai

Network ID = 132.92.121
Host ID = 1

Dengan kata lain, 132.92.121.1 ialah host nomor 1, pada jaringan 132.92.121/24.

Berkat subnet mask baru ini pula, satu jaringan kelas B ini akhirnya menjadi 256 jaringan baru: 132.92.0.xxx, 132.92.1.xxx, 132.92.2.xxx, hingga 132.92.255.xxx (biasa dituliskan sebagai 132.92.0/16, 132.92.1/16, 132.92.2/16 dan seterusnya).

Dengan teknik di atas, Anda mengalokasikan IP address kelas B menjadi sekian banyak network yang **berukuran sama**.

Untuk memperjelas hal di atas, berikut ini diberikan beberapa contoh IP address beserta subnet mask dan artinya.

Tabel 3.3 Beberapa contoh subnet mask dan artinya

IP Address	Subnet Mask	Network Address	Broadcast Address	Interpretasi
44.132.1.20	255.255.0.0 (16 bit)	44.132.0.0 (kelas A)	44.132.255.255	Host 1.20 pada subnet 44.132.0.0
44.132.1.20	255.255.255.0 (24 bit)	44.132.1.0 (kelas A)	44.132.1.255	Host 20 pada subnet 44.132.1.0
81.150.2.3	255.255.255.0 (24 bit)	81.150.2.0 (kelas A)	81.50.2.255	Host 3 pada subnet 81.150.2.0

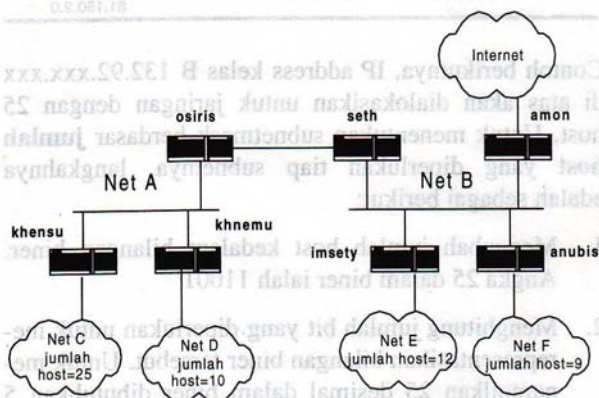
Contoh berikutnya, IP address kelas B 132.92.xxx.xxx di atas akan dialokasikan untuk jaringan dengan 25 host. Untuk menentukan subnetmask berdasar **jumlah host** yang diperlukan tiap subnetnya, langkahnya adalah sebagai berikut:

1. Mengubah jumlah host kedalam bilangan biner. Angka 25 dalam biner ialah 11001

2. Menghitung jumlah bit yang diperlukan untuk merepresentasikan bilangan biner tersebut. Untuk menampilkan 25 desimal dalam biner dibutuhkan 5 bit. 5 bit ialah jumlah bit yang dibutuhkan untuk host-ID. Dengan demikian, panjang bit subnet-ID ialah selisih antara panjang host ID lama dan host-ID baru. Jika IP yang digunakan ialah kelas B (dengan host ID default 16 bit) maka panjang bit subnet ID ialah 16 – 5 = 9 bit. Dengan demikian subnet masknya ialah:

   255.255.255.224 =
   11111111.11111111.11111111.111**00000**

3.4.7. Contoh 2: Implementasi subnetting untuk alokasi IP address

Contoh jaringan di bawah akan digunakan untuk menerangkan konsep subnetting. Konfigurasi jaringan yang mirip dengan ini nantinya juga akan digunakan untuk menerangkan konsep IP routing. Hanya saja, alokasi IP yang digunakan berbeda. Alokasi IP yang dipakai saat ini sengaja dipilih untuk mempermudah pemahaman konsep subnetting. Sedangkan alokasi IP yang digunakan untuk menerangkan IP routing telah dipilih dan dioptimasi untuk keperluan tersebut.

Gambar 3.7 Contoh jaringan untuk implementasi subnet.

Pada Gambar 3.7 terdapat sebuah network besar yang berisi 7 router dan 7 network kecil. Jaringan di atas terdiri atas dua network utama, yaitu A dan B, yang terletak di lokasi yang berbeda. Network A terhubung ke network B dan ke internet melalui router *osiris* dan *seth,* menggunakan jalur WAN (misalnya frame relay

atau X.25). Kondisi jaringan yang mirip dengan ini bisa dijumpai pada WAN perusahaan dengan kantor pusat (network B) dan satu kantor cabang (network A).

Untuk keperluan alokasi IP, telah disediakan IP address kelas B 132.92.xxx.xxx.

Dalam melakukan subnetting, kita harus terlebih dahulu menentukan seberapa besar jaringan kita saat ini, serta kemungkinan perkembangannya di masa mendatang. Untuk menentukannya, Anda dapat mengikuti petunjuk umum berikut:

1. Tentukan dulu jumlah jaringan fisik yang ada
2. Tentukan jumlah IP address yang dibutuhkan oleh masing-masing jaringan tersebut
3. Berdasarkan requirement ini, definisikan:
   - satu subnet mask untuk seluruh network
   - subnet ID yang unik untuk tiap segmen jaringan
   - range Host-ID untuk tiap subnet

Menentukan jumlah subnetwork dan subnet ID

Langkah pertama yang dilakukan ialah menghitung jumlah subnetwork yang terdapat pada Gambar 3.6. Berdasarkan gambar, ada tujuh network, yaitu A, B, C, D, E, dan F, serta network antara router *osiris* dan *seth*. Dengan demikian, diperlukan tujuh subnet ID.

Menentukan jumlah host ID tiap subnet

Langkah berikutnya ialah menentukan alokasi jumlah host-ID yang dibutuhkan oleh masing-masing subnet.

Host-ID diberikan satu buah untuk tiap host dan satu buah untuk tiap interface dari router.

Network A

Network A berisi 3 buah router, yaitu *khensu*, *khnemu*, dan *osiris*. Dengan demikian network A membutuhkan 3 buah host ID

Network B

Network B terdiri atas 4 buah router: *seth*, *ammon*, *imsety*, dan *anubis*. Network ini membutuhkan 4 buah host ID

Network C, D, E, dan F

Jumlah host ID yang dibutuhkan keempat subnet di atas ialah jumlah host masing-masing subnet ditambah satu untuk router. Dengan demikian, subnet C membutuhkan 26 host ID, D membutuhkan 11 host ID, E dan F masing-masing membutuhkan 13 dan 10 host ID.

Jalur antara *osiris* dan *seth*

Jalur antara router *osiris* dan *seth* membutuhkan 2 host ID.

Menentukan subnet mask, subnet-ID, dan range IP address untuk tiap subnet

Cara termudah dalam menentukan subnet mask ialah menyamaratakan subnet mask seluruh subnet dengan menggunakan subnet mask milik subnet dengan jumlah host terbesar. Cara ini cocok untuk digunakan jika:

- mendapat alokasi IP yang besar sekali
- menggunakan IP private (dan terkoneksi ke internet melalui proxy server)

- masih menggunakan protokol routing RIP versi 1. Penjelasan lebih lanjut dapat ditemui pada Bab 5 mengenai IP routing.

Jika jaringan Anda tidak memenuhi salah satu dari ketiga syarat di atas, jaringan sebaiknya dirancang dengan alokasi IP yang efisien. Efisien disini memiliki dua arti, yaitu efisien dalam menggunakan IP dan efisien dalam menempatkan IP sehingga dihasilkan tabel routing yang kecil (akibat proses agregasi tabel routing). Untuk melakukan hal ini, digunakan teknik yang disebut sebagai *Variable Length Subnet Mask* (VLSM). Teknik VSLM akan diterangkan setelah subbab ini.

Jika digunakan pendekatan dengan menggunakan ukuran subnet yang sama, caranya mudah. Mula mula dicari subnet dengan kebutuhan host ID terbesar. Berdasar informasi pada gambar 3.7, subnet C memiliki kebutuhan host-ID terbesar yaitu 26 host-ID. Jika angka ini dikonversikan dalam biner, akan dihasilkan: 11010

Untuk merepresentasikan angka 26 dalam biner, diperlukan 5 bit. Artinya, untuk host–ID, kita membutuhkan tempat sebanyak 5 bit. Dengan demikian, subnet mask yang digunakan ialah:

11111111 . 11111111 . 11111111 . 11100000 =
255.255.255.224

Kemudian, berdasarkan blok IP yang didelegasikan kepada Anda, Anda dapat mengalokasikan IP untuk tiap subnet sesuai dengan subnet mask di atas. Alokasi IP untuk tiap subnet dilakukan dengan membuat daftar -subnet ID. Caranya ialah:

1. Mengisi bit host-ID dengan 0
2. Menuliskan seluruh kombinasi bit pada daerah bit subnet-ID
3. Mengubah bit subnet id menjadi desimal.

Jika Anda menggunakan netmask 255.255.255.224 untuk IP address kelas B 132.92.xxx.xxx, daftar subnet-IDnya terdapat pada Gambar 3.8

Gambar 3.8 Subnet ID untuk netmask 255.255.255.224

Berdasarkan Gambar 3.8, subnet yang terbentuk ialah:

132.92.0.0/27
132.92.0.32/27
132.92.0.64/27
dan seterusnya.

Sesuai dengan aturan dasar subnetting dari RFC 950, subnet dengan subnet-ID 0 tidak boleh digunakan. Demikian pula dengan subnet-ID 255. Hal ini dilakukan untuk mencegah kesalahan pembacaan subnet oleh protokol routing RIP Versi 1. Dalam mengirimkan

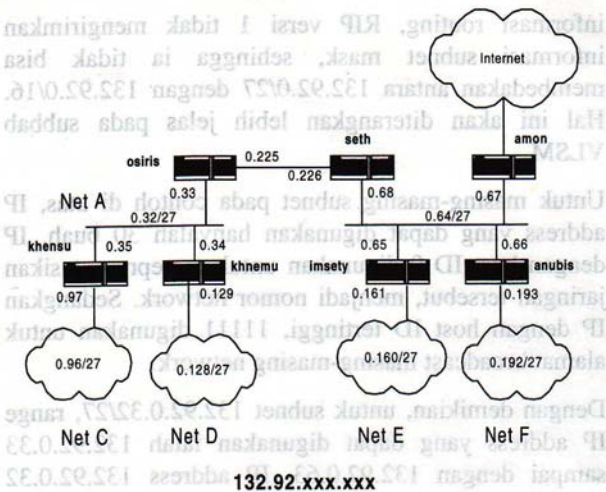
informasi routing, RIP versi 1 tidak mengirimkan informasi subnet mask, sehingga ia tidak bisa membedakan antara 132.92.0/27 dengan 132.92.0/16. Hal ini akan diterangkan lebih jelas pada subbab VLSM.

Untuk masing-masing subnet pada contoh di atas, IP address yang dapat digunakan hanyalah 30 buah. IP dengan host ID 0 digunakan untuk merepresentasikan jaringan tersebut, menjadi nomor network. Sedangkan IP dengan host ID tertinggi, 11111 digunakan untuk alamat broadcast masing-masing network.

Dengan demikian, untuk subnet 132.92.0.32/27, range IP address yang dapat digunakan ialah 132.92.0.33 sampai dengan 132.92.0.63. IP address 132.92.0.32 digunakan sebagai alamat network dan 132.92.0.63 adalah alamat broadcast subnet tersebut. Hal yang sama juga berlaku untuk subnet subnet lainnya, seperti yang tercantum pada Tabel 3.4

Tabel 3.4 Daftar subnet yang dialokasikan

Alamat Subnet	Alamat broadcast	Range IP address
132.92.0.32	132.92.0.63	132.92.0.33 s.d. 132.92.0.62
132.92.0.64	132.92.0.95	132.92.0.65 s.d. 132.92.0.94
132.92.0.96	132.92.0.127	132.92.0.97 s.d. 132.92.0.126
132.92.0.128	132.92.0.159	132.92.0.129 s.d. 132.92.0.158
132.92.0.160	132.92.0.191	132.92.0.161 s.d. 132.92.0.190
132.92.0.192	132.92.0.223	132.92.0.193 s.d. 132.92.0.222
132.92.0.224	132.92.0.255	132.92.0.225 s.d. 132.92.0.254

Berdasarkan Tabel 3.4 di atas, network pada Gambar 3.7 memiliki alokasi IP sebagai berikut:

Gambar 3.9 Alokasi IP di jaringan

3.4.8. Variable Length Subnet Mask (VLSM)

Masalah dengan subnet sama rata

Ada baiknya saat ini Anda melihat kembali pengalokasian IP address yang dilakukan sebelumnya, yang terdapat pada Gambar 3.9. Di sana terlihat beberapa kekurang sempurnaan.

Pada contoh di atas, subnet yang kecil mendapat alokasi IP yang sama dengan subnet yang besar. Bahkan subnet yang hanya berisi dua host saja mendapat alokasi IP yang sama dengan subnet berisi 25 host. Hal ini merupakan pemborosan IP, suatu hal yang harus dihindari di jaman ketika IP address sulit didapat, seperti saat ini.

Namun, hal di atas tak bisa dihindari ketika Anda menggunakan protokol routing RIP versi 1. Ketika protokol routing ini digunakan, subnet mask yang dipakai di seluruh jaringan harus seragam. Hal ini terpaksa dilakukan karena RIP versi 1 tidak mengirimkan subnet mask sebagai bagian dari informasi routingnya. Karena ketiadaan informasi ini, RIP dipaksa membuat asumsi sederhana tentang subnet mask yang seharusnya ia gunakan pada **seluruh route** yang ia ketahui.

Asumsi yang digunakan RIP versi 1 dalam menentukan subnet mask dari suatu rute ialah:

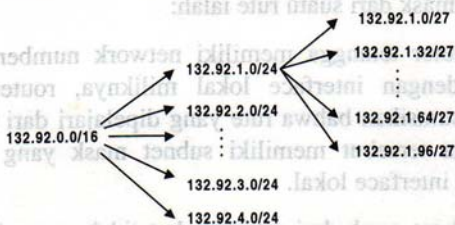
Jika router tetangga memiliki network number yang sama dengan interface lokal miliknya, router kita mengasumsikan bahwa rute yang dipelajari dari router tetangga tersebut memiliki subnet mask yang sama dengan interface lokal.

Jika subnet mask dari rute tersebut tidak sama dengan milik interface lokal, router berasumsi netwok tersebut tidak memiliki subnet mask dan memberikan subnet mask asli dari IP address tersebut.

Karenanya, langkah yang harus dilakukan agar suatu network dapat berjalan dengan subnet mask yang berbeda-beda, yaitu mengganti protokol routing RIP versi 1 tersebut dengan protokol routing lain, seperti RIP versi 2 atau OSPF.

Jaringan yang menerapkan ukuran subnet yang berbeda-beda ini tentu saja menerapkan subnet mask yang berbeda-beda (variabel) untuk tiap subnet nya. Hal ini disebut sebagai *Variable Length Subnet Mask* (VLSM).

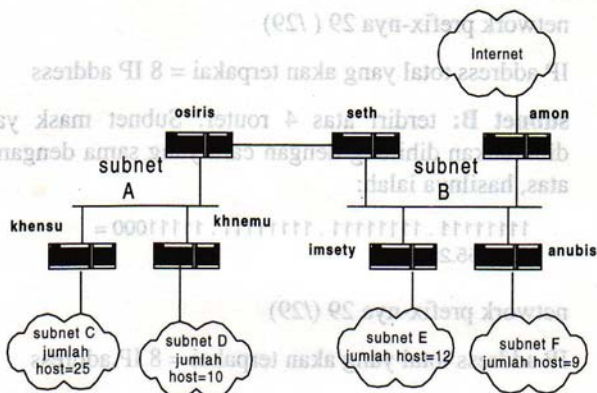
Agregasi Routing

VLSM memungkinkan dibaginya ruang IP address secara rekursif sehingga dapat disusun kembali di level paling atas untuk mengurangi jumlah informasi routing. Secara konsep, sebuah network mula-mula dibagi menjadi subnet, yang kemudian dibagi lagi menjadi subsubnet, untuk kemudian dibagi lagi menjadi subsubsubnet, dan seterusnya. Hal ini memungkinkan struktur informasi routing secara detail dari satu subnet disembunyikan dari router lain pada subnet yang lain.

Gambar 3.10 Agregasi routing

Dengan teknik ini pula, subnet yang tidak memerlukan alokasi IP besar dapat mengambil alokasi IP yang kecil.

3.4.9. Contoh 3: Implementasi Subnetting dengan VLSM

Untuk memperjelas masalah dan pemecahannya, kita akan langsung mencoba mengalokasikan IP address dengan metode VLSM. Jaringan yang digunakan sebagai contoh sama dengan sebelumnya. Demikian pula ruang IP address yang disediakan, yaitu

132.92.xxx.xxx

Gambar 3.11 Network untuk implementasi RIPv2

Tujuan desain alokasi IP kita kali ini ialah:

- Menggunakan IP address sehemat mungkin.
- Menyembunyikan detail routing jaringan terhadap network luar

Langkah-langkah yang digunakan ialah sebagai berikut:

Menentukan subnet mask untuk tiap subnet berdasarkan jumlah host

subnet A: terdiri atas tiga router (*khensu, osiris, dan khnemu)*. Untuk itu dibutuhkan tiga host ID. Selain untuk host ID, diperlukan juga alamat broadcast dan alamat network untuk subnet ini, sehingga dibutuhkan lima alamat. Angka 5 dalam biner ialah 101, terdiri atas 3 bit. Dengan demikian kita membutuhkan 3 bit untuk menuliskan host ID. Subnet mask yang dibutuhkan ialah:

11111111 . 11111111 . 11111111 . 11111000 = 255.255.255.248

network prefix-nya 29 (/29)

IP address total yang akan terpakai = 8 IP address

subnet B: terdiri atas 4 router. Subnet mask yang dibutuhkan dihitung dengan cara yang sama dengan di atas, hasilnya ialah:

11111111 . 11111111 . 11111111 . 11111000 = 255.255.255.248

network prefix-nya 29 (/29)

IP address total yang akan terpakai = 8 IP address

subnet C: terdiri atas 25 host + router (*khensu*). Untuk merepresentasikan 26 host ditambah alamat network dan broadcast dibutuhkan 5 bit. Sehingga subnet mask yang dihasilkan ialah:

11111111 . 11111111 . 11111111 . 11100000 = 255.255.255.224

network prefix-nya 27 (/27)

IP address total yang akan terpakai = 32 IP address

subnet D: terdiri atas 10 host + router (*khnemu*). Untuk merepresentasikan 11 host ditambah alamat network dan broadcast dibutuhkan 4 bit. Subnet mask yang dihasilkan ialah:

11111111 . 11111111 . 11111111 . 11110000 = 255.255.255.240

network prefix-nya 28 (/28)

IP address total yang akan terpakai = 16 IP address

subnet E: subnet mask yang dihasilkan sama dengan subnet D, yaitu:

11111111 . 11111111 . 11111111 . 11110000 = 255.255.255.240

network prefix-nya 28 (/28)

IP address total yang akan terpakai = 16 IP address

subnet F: subnet mask yang dihasilkan sama dengan subnet D, yaitu:

11111111 . 11111111 . 11111111 . 11110000 = 255.255.255.240

network prefix-nya 28 (/28)

IP address total yang akan terpakai = 16 IP address

Subnet antara *Osiris* dan *Seth:* Terdiri atas dua host . Untuk merepresentasikan dua host ditambah alamat network dan broadcast dibutuhkan 2 bit. Subnet mask yang dihasilkan:

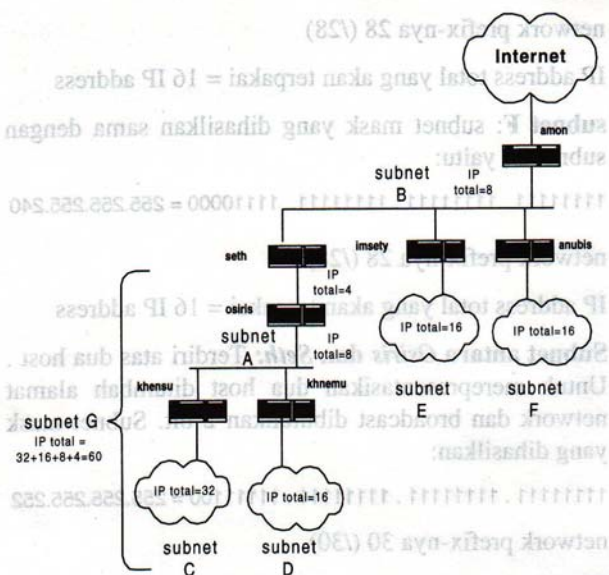
11111111 . 11111111 . 11111111 . 11111100 = 255.255.255.252

network prefix-nya 30 (/30)

IP address total yang terpakai = 4 IP address

Menggambar ulang diagram jaringan, menyerupai diagram akar

Hasil penggambaran ulang diagram jaringan seperti ditujukkan Gambar 3.12.

Usaha penggambaran ulang ini dilakukan untuk melihat network-network yang dapat digabung untuk diberi satu blok IP. Ukuran blok IP ditentukan dari jumlah host yang terkandung dalam satu subnet. Penggabungan ini memerlukan kehati-hatian agar jika dilihat dari luar subnet, sedapat mungkin hanya terlihat satu network prefix saja.

Gambar 3.12

Contoh yang paling jelas ialah subnet G di atas, yang terdiri atas subnet A, C, D, dan *osiris-seth*. Penggabungan beberapa subnet ini menjadi subnet G diperlukan agar jika dilihat dari luar subnet G hanya terlihat satu subnet saja, sama dengan melihat network di bawah router *imsety* dan *anubis* (network E dan F).

Dengan jumlah IP total yang mencapai 60 buah (32 dari subnet C + 16 dari subnet D + 8 dari subnet A + 4 dari *seth-osiris*), subnet G memerlukan bit host ID 6 bit (2^6 = 64). Subnet mask untuk subnet ini jika dilihat dari luar ialah:

11111111 . 11111111 . 11111111 . 11000000 = 255.255.255.192.

Karena subnet mask di luar network G dan didalam network G berbeda (Variabel), maka teknik alokasi IP ini disebut sebagai **Variable Length Subnet Mask** (VLSM).

Membuat tabel alokasi IP berdasarkan diagram jaringan.

Berdasarkan diagram jaringan pada Gambar 3.12, dibuatlah tabel alokasi IP seperti pada Tabel 3.5. Bagian kolom dari tabel ini menunjukkan jumlah bit dalam subnet mask, panjang network prefix, diikuti netmask dalam desimal serta byte terakhir netmask dalam bentuk biner. Setiap kotak dalam tabel melambangakan satu subnet, dengan nomor network di ujung kiri atas.

Dari Tabel 3.5 terlihat bahwa jumlah IP yang dibutuhkan kurang dari 1 kelas C. Hal ini hanya dimungkinkan jika kita menggunakan VLSM. Dengan demikian, Anda bisa memilih menempatkan seluruh alokasi ini pada bagian kelas C yang mana saja dari 132.92.xxx.xxx (misalnya 132.92.1.xxx, 132.92.2.xxx dan seterusnya)

Misalkan Anda memilih alokasi kelas C di 132.92.122.xxx, maka daftarnya seperti dalam Tabel 3.6.

Tabel 3.5 Alokasi IP jaringan

2 bit (/26) 255.255.255.192 11000000	3 bit (/27) 255.255.255.224 11100000	4 bit (/28) 255.255.255.240 11110000	5 bit(/29) 255.255.255.248 11111000	6 bit (/30) 255.255.255.252 11111100
0	0	0 subnet E	0	0
				4
			8	8
				12
		16 subnet F	16	16
				20
			24	24
				28
	32	32	32 subnet B	32
				36
			40	40
				44
		48	48	48
				52
			56	56
				60
64 subnet G	64 subnet C	64	64	64
				68
			72	72
				76
		80	80	80
				84
			88	88
				92
	96	96 subnet D	96	96
				100
			104	104
				108
		112	112 subnet A	112
				116
			120	120 seth-osiris
				124

Tabel 3.6 Alokasi IP kelas C 132.92.122/24

Subnet	Nomor Network	Alamat Broadcast	Range IP address
Subnet E	132.92.122.0/28	132.92.122.15	132.92.122.1 s.d. 132.92.122.14
Subnet F	132.92.122.16/28	132.92.122.31	132.92.122.17 s.d. 132.92.122.30
Subnet B	132.92.122.32/29	132.92.122.39	132.92.122.33 s.d. 132.92.122.38
Subnet C	132.92.122.64/27	132.92.122.95	132.92.122.65 s.d. 132.92.122.94
Subnet D	132.92.122.96/28	132.92.122.111	132.92.122.97 s.d. 132.92.122.110
Subnet A	132.92.122.112/29	132.92.122.119	132.92.122.113 s.d. 132.92.122.118
Seth-osiris	132.92.122.120/30	132.92.122.123	132.92.122.121 s.d. 132.92.122.122

3.5. Ringkasan

IP address merupakan pengenal yang digunakan untuk memberi alamat suatu host di internet. Format IP address ialah bilangan 32 bit yang tiap 8 bitnya dipisahkan oleh tanda titik. Untuk mempermudah distribusinya, IP address dibagi dalam kelas kelas: A, B,C, D, dan E. Kelas A berjumlah sedikit namun bisa diisi banyak host. Kelas C berjumlah banyak namun berisi sedikit host.

IP address terdiri atas dua bagian, yaitu network ID dan host ID. Network ID menunjukkan nomor network, sedangkan host ID mengidentifikasi host dalam satu network. Host ID harus unik dalam satu network.

Untuk mengefisienkan alokasi IP address, dilakukan subnetting. Subnetting ialah proses memecah satu kelas IP address menjadi beberapa subnet dengan jumlah host yang lebih sedikit. Untuk menentukan batas network ID dan host ID dalam suatu subnet, digunakan subnet mask.

Cara paling sederhana dalam membentuk subnet ialah mengalokasikan IP address sama rata untuk tiap subnet. Namun hal ini hanya cocok jika alokasi IP yang kita miliki besar sekali atau kita menggunakan IP privat, dan jaringan menjalankan protokol routing RIP versi 1. Jika kita ingin membuat jaringan dengan subnet berukuran berbeda, RIP versi 1 tidak dapat digunakan.

Alokasi IP dengan subnet yang besarnya berbeda-beda sesuai kebutuhan disebut sebagai VLSM (*Variable Length Subnet Mask*). VLSM dapat menghasilkan alokasi IP yang lebih efisien.

Bab 4
Domain Name System

4.1. Mengapa Harus Menggunakan DNS

Dengan berjalannya routing paket IP, *Wide Area Network* berbasis protokol TCP/IP sudah dapat diwujudkan. Namun dalam penggunaannya, ternyata muncul lagi masalah baru. Saat pengguna aplikasi Internet hendak menghubungi komputer tujuan, dia harus menyebutkan terlebih dulu IP address dari komputer tujuan tersebut.

Padahal, jika dilihat dari formatnya yang berupa angka, dapat diduga bahwa IP address cukup sulit diingat. Lebih mudah bagi manusia untuk mengingat nama dari pada angka. Karenanya, agar Internet lebih mudah digunakan, diperlukan suatu cara untuk memetakan dari IP address ke nama host/komputer dan sebaliknya.

Pada awalnya, digunakan teknik yang dinamakan *host table*. Masing-masing host/komputer menyimpan daftar kombinasi nama komputer dan IP address, pada file yang dinamakan HOSTS.TXT. file ini berisi nama dan *IP address* ***seluruh komputer yang terkoneksi ke Internet***. File ini pula yang setiap kali diperbarui

melalui FTP ke seluruh host di Internet, jika terdapat penambahan host baru.

Saat Internet masih merupakan komunitas yang kecil, hal ini tidak menjadi masalah. Masalah baru timbul ketika Internet semakin membesar dan membesar. Tahun 1984, jumlah komputer yang terhubung ke Internet mencapai 1000 buah. Jumlah host yang harus ditulis di HOSTS.TXT menjadi terlalu banyak dan cara ini pun menjadi tidak efisien.

Karenanya, muncul ide untuk melakukan pendistribusian database *hostname* dan IP address ini. Dengan pendistribusian ini, masing-masing organisasi hanya bertanggung jawab terhadap database yang berisi informasi jaringan miliknya saja. Karena sifat database yang terdistribusi, maka harus ada suatu mekanisme bagi host lain untuk bisa menemukan host yang tepat, yang menyimpan data yang ia butuhkan.

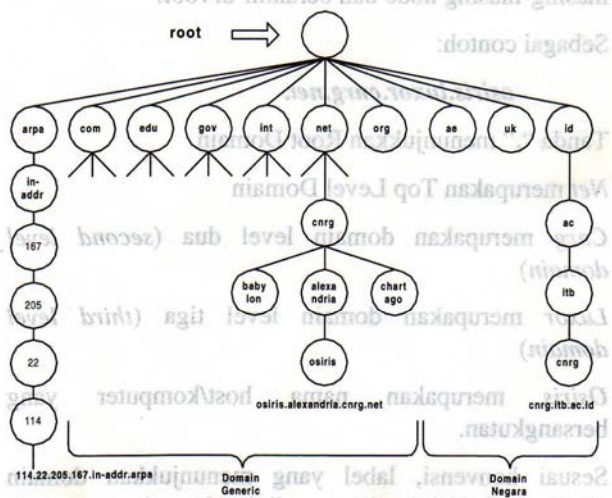
Tahun 1984, **Paul Mockapetris** mengusulkan sistem database terdistribusi yang dinamakan DNS (*Domain Name System*). Sistem inilah yang digunakan hingga sekarang.

Selain untuk memetakan IP address dan Nama Host, DNS juga digunakan sebagai sarana bantu penyampaian e-mail (*e-mail routing*). Hal ini akan diterangkan pada subbab konfigurasi DNS.

4.2. Top Level Domain dan Pendelegasian

Untuk membentuk sistem database yang terdistribusi, yang pertama kali harus diatur ialah format data.

Format data ini harus dibuat sedemikian rupa, sehingga ia cocok digunakan untuk sistem yang terdistribusi. Karena informasi yang hendak disimpan ialah IP address dan nama host, maka format penamaan host harus konsisten untuk semua host dan mampu mencerminkan terdistribusinya data tersebut.

Format penamaan host di Internet dibuat memiliki hirarki. Skema hirarki tersebut digambarkan berbentuk *tree*. Satu node/titik membentuk *tree*, memiliki beberapa subnode. Subnode ini membentuk *tree* yang memiliki beberapa subnode lagi, dan seterusnya. Pada masing-masing node ini terdapat label. Node berlabel ini disebut *domain*. Domain ini bisa berupa nama host, subdomain atau top level domain.

Gambar 4.1 Domain name space

Domain teratas ialah Root Domain. Domain ini dituliskan dalam bentuk titik("."). Top Level Domain terdiri atas semua node yang tepat berada di bawah root. Pada gambar di atas hal ini ditunjukkan dengan node **arpa,com,edu,gov** dan seterusnya

Subdomain merupakan kumpulan *keturunan* Top level Domain. Node yang berada tepat di bawah *Top Level Domain* disebut *Second Level Domain.* Node di bawah *Second Level Domain* disebut *Third Level Domain*, dan seterusnya.

Cara pembentukan serta pembacaan nama host dan domain, sesuai dengan diagram di atas, dimulai dari node paling bawah, mengikuti label yang tertera pada masing-masing node dan berakhir di *root.*

Sebagai contoh:

osiris.luxor.cnrg.net.

Tanda "." menunjukkan Root Domain

Net merupakan Top Level Domain

Cnrg merupakan domain level dua (*second level domain*)

Luxor merupakan domain level tiga (*third level domain*)

Osiris merupakan nama host/komputer yang bersangkutan.

Sesuai konvensi, label yang menunjukkan domain ditulis dari kiri ke kanan, dipisahkan dengan tanda titik, dengan domain yang paling jauh dari root ditulis terlebih dahulu. Penulisan secara lengkap seperti di atas, mulai dari nama host hingga tanda titik yang

melambangkan root disebut sebagai Fully Qualified Domain Name (FQDN).

Top level domain digunakan untuk menunjukan jenis perusahaan, instansi, lembaga, atau negara tempat komputer ini berada. Top level domain (TLD) ini dapat dibagi menjadi 3 jenis, yaitu:

- TLD generik (*generic domain*)
- TLD negara (*country domain*)
- TLD arpa.

Pada mulanya TLD yang dipakai ialah TLD generik. TLD generik ini terdiri atas tujuh jenis domain yang terdiri atas tiga huruf. Domain **com**, digunakan oleh organisasi bersifat komersial (*ibm.com, microsoft.com*). Domain **edu** (*berkeley.edu.,purdue.edu.,mit.edu.*), digunakan untuk lembaga pendidikan (universitas). Domain **gov**, digunakan untuk lembaga pemerintahan Amerika Serikat (*whitehouse.gov., odci.gov.*). Domain **int** digunakan oleh organisasi internasional (*nato.int.*). Domain **mil** digunakan oleh badan kemiliteran Amerika Serikat (*army.mil., navy.mil., af.mil.*). Domain **net** digunakan oleh penyedia jaringan Internet (*ibm.net, mci.net.*). Sedangkan domain **org** digunakan oleh organisasi nonkomersial (*greenpeace.org*). Hingga saat ini sistem pembagian organisasional seperti ini masih berlaku di Amerika.

Berdasarkan informasi ini kita dapat menyimpulkan bahwa komputer *osiris.luxor.cnrg.net.* di atas ialah komputer milik organisasi yang menyediakan jasa jaringan intenet.

Dengan semakin banyaknya negara-negara yang terhubung ke Internet, kemudian diputuskan untuk menggunakan standard pembagian geografis yang ditetapkan sesuai standar ISO 3166. Inilah yang disebut TLD Negara (*Country Domain*). Berdasarkan konvensi tersebut, dialokasikan Top Level Domain yang merupakan pengenal geografis (negara) dan terdiri atas dua huruf yang unik.

Sebagai contoh, negara Indonesia memiliki top level domain **.id,** negara Inggris memiliki top level domain **.uk** (United Kingdom), negara Malaysia memiliki top level domain **my**, dan sebagainya.

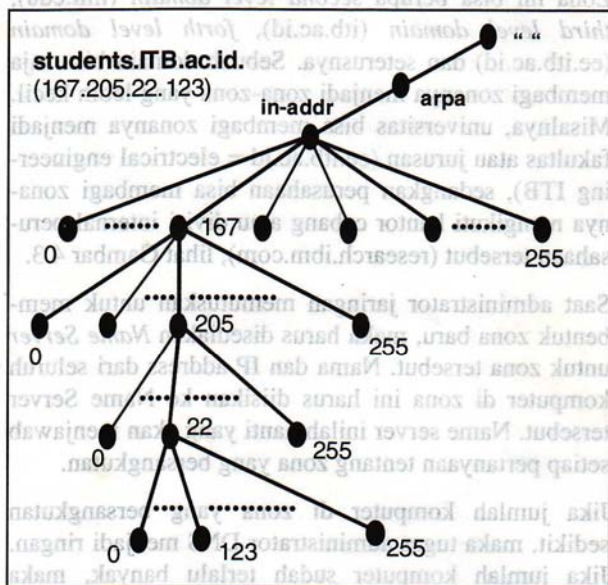
Pada umumnya pembagian TLD secara geografis ini kemudian diikuti dengan pembagian berdasarkan afiliasi organisasi bagi level domain di bawahnya. Ada yang mengadopsi sistem pembagian di Amerika, seperti **edu.au** atau **com.au**, ada juga yang mengikuti sistem pembagian yang dipelopori Inggris, seperti **co.uk** (*corporation*), atau **ac.uk** (*academic*).

Konvensi pembagian nama domain di Indonesia ditetapkan sampai level kedua (*Second Level Domain*). Sedangkan aturan penamaan domainnya mengikuti sistem inggris. Hal ini ditunjukkan pada Tabel 4.1 sebagai berikut:

Tabel 4.1 Second Level Domain untuk Indonesia

Domain	Keterangan
go.id	Subdomain untuk lembaga pemerintah
co.id	Subdomain untuk lembaga konvensional
ac.id	Institusi akademik
net.id	Penyedia jasa network
or.id	LSM dan lembaga non komersial

Untuk penamaan *third level domain* dan seterusnya, hal ini diserahkan pada pengelola jaringan yang bersangkutan. Misalkan Anda pengelola jaringan komputer di "Universitas Atas Angin", Anda dapat menggunakan domain **atasangin.ac.id**, **angin.ac.id** dan sebagainya, sepanjang domain ini belum dimiliki oleh universitas lain.

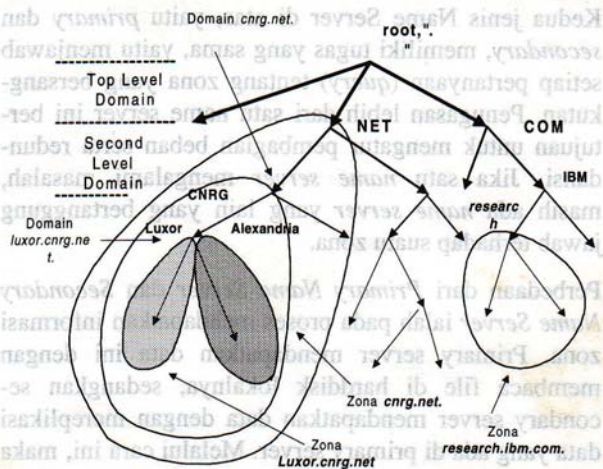
Selain memetakan nama host ke IP address, DNS juga memiliki fasilitas *reverse mapping*. Reverse mapping ialah proses memetakan IP address ke *domain-name*. Reverse mapping juga digunakan untuk menghasilkan keluaran yang lebih manusiawi, mudah dibaca dan diinterpretasikan, misalnya untuk pembacaan *log-file*.

Gambar 4.2 Reverse Address Tree

Saat suatu organisasi bergabung ke Internet dan mendapatkan otoritas untuk nama domain tertentu, dia juga mendapat otoritas untuk *namespace* **in-addr.arpa**, yang sesuai dengan IP address yang dimilikinya.

Keseluruhan *domain name space* di atas, baik yang biasa maupun yang *reverse mapping*, sebagaimana digambarkan di Gambar 4.1 dan 4.2 tidak dikelola oleh satu server. Pengelolaannya dilakukan secara terdistribusi. Untuk itu, domain name space di atas dibagi kedalam zone zone. Sebuah zone meliputi seluruh host di bawah domain tertentu, kecuali yang didelegasikan ke zona lain.

Zona ini bisa berupa *second level domain* (mit.edu), *third level domain* (itb.ac.id), *forth level domain* (ee.itb.ac.id) dan seterusnya. Sebuah domain bisa saja membagi zonanya menjadi zona-zona yang lebih kecil. Misalnya, universitas bisa membagi zonanya menjadi fakultas atau jurusan (ee.itb.ac.id = electrical engineering ITB), sedangkan perusahaan bisa membagi zonanya mengikuti kantor cabang atau divisi internal perusahaan tersebut (research.ibm.com), lihat Gambar 4.3.

Saat administrator jaringan memutuskan untuk membentuk zona baru, maka harus disediakan *Name Server* untuk zona tersebut. Nama dan IP address dari seluruh komputer di zona ini harus diisikan ke Name Server tersebut. Name server inilah nanti yang akan menjawab setiap pertanyaan tentang zona yang bersangkutan.

Jika jumlah komputer di zona yang bersangkutan sedikit, maka tugas administrator DNS menjadi ringan. Jika jumlah komputer sudah terlalu banyak, maka sebaiknya dilakukan pembentukan zona baru serta pendelegasian domain.

Gambar 4.3 Domain dan zona

Jika pada suatu domain ingin dibentuk zona baru, ditugaskan sebuah *Primary Name Server* dan satu atau lebih *Secondary Name Server* untuk menangani zona ini. Dengan demikian, Server DNS induk telah memberikan *authority* pada name server yang bersangkutan untuk menangani zona tersebut. Kedua jenis server ini pun menjadi Name Server yang *authoritative* untuk zona baru tersebut.

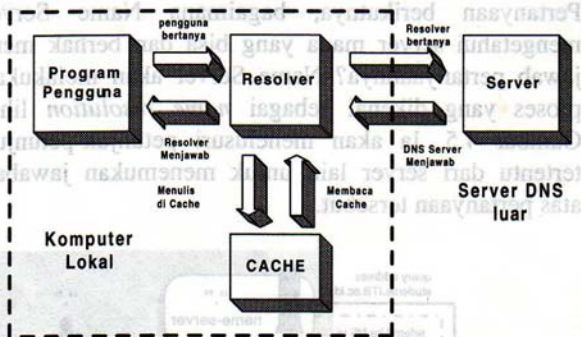
Sebagai konsekwensinya, data zona yang terdelegasi pada Name Server domain induk hanya berisi penunjuk ke sejumlah name-server yang *authoritative* untuk zona ini, bukan berisi data/informasi dari zona yang didelegasikan. Dengan demikian, bila suatu saat terdapat pertanyaan terhadap data-data tersebut, server akan menjawab memberikan daftar name-server yang dapat dan "lebih patut" untuk ditanya.

Kedua jenis Name Server di atas, yaitu *primary* dan *secondary*, memiliki tugas yang sama, yaitu menjawab setiap pertanyaan (*query*) tentang zona yang bersangkutan. Penugasan lebih dari satu name server ini bertujuan untuk mengatur pembagian beban serta redundansi. Jika satu *name server* mengalami masalah, masih ada *name server* yang lain yang bertanggung jawab terhadap suatu zona.

Perbedaan dari *Primary Name Server* dan *Secondary Name Server* ialah pada proses mendapatkan informasi zona. Primary server mendapatkan data ini dengan membaca file di harddisk lokalnya, sedangkan secondary server mendapatkan data dengan mereplikasi data yang ada di primary server. Melalui cara ini, maka proses pembaruan data DNS menjadi mudah. Data hanya perlu di perbarui di primary server saja. Secondary server akan memperbarui datanya melalui proses replikasi ini. Data di *primary server* disebut *Zona File*, sedangkan proses ini pembaruan data pada *secondary server* disebut sebagai *Zona Transfer*.

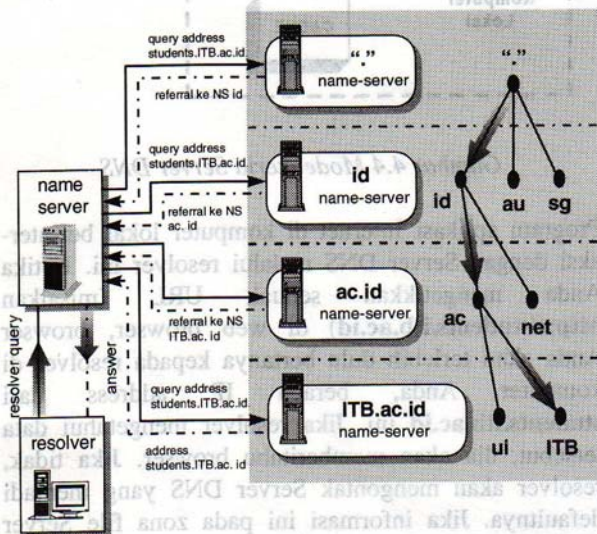
4.3. Komponen DNS

Untuk memahami cara kerja DNS, maka kita terlebih dulu harus mengetahui komponen-komponennya. Seperti yang ditunjukkan pada Gambar 4.4.

Resolver ialah bagian dari program aplikasi yang berfungsi menjawab pertanyaan program aplikasi tentang domain. Resolver akan menjawab pertanyaan dengan dua cara, yaitu melihat isi *cache*nya (dengan asumsi bahwa pertanyaan ini sudah pernah ditanyakan dan jawabannya tersimpan di *cache*), dan bertanya kepada Server DNS serta menginterpretasikan hasilnya.

Gambar 4.4 Model kerja Server DNS

Program aplikasi Internet di komputer lokal berinteraksi dengan Server DNS melalui resolver ini. Ketika Anda mengetikkan sebuah URL (misalkan **http://students.itb.ac.id**) di web browser, browser Anda akan terlebih dulu bertanya kepada resolver di komputer Anda, berapa IP address dari **students.itb.ac.id** ini. Jika resolver mengetahui data tersebut, dia akan memberitahu browser. Jika tidak, resolver akan mengontak Server DNS yang menjadi defaultnya. Jika informasi ini pada zona file Server DNS, ia akan segera menjawab dan browser pun mendapat informasi yang benar.

Bagaimana jika Name Server tidak mengetahui jawabnya (atau name server tidak *authoritative* untuk zona tersebut)? Name server ini akan berusaha mencari jawabnya, dengan bertanya kepada Name Server lain yang mengetahuinya (yang authoritative).

Pertanyaan berikutnya, bagaimana Name Server mengetahui server mana yang bisa dan berhak menjawab pertanyaannya? Name Server akan melakukan proses yang dikenal sebagai *name resolution* lihat Gambar 4.5. Ia akan menelusuri petunjuk-petunjuk tertentu dari server lain untuk menemukan jawaban atas pertanyaan tersebut.

Gambar 4.5 Name Resolution

Untuk mengetahuinya, name server kita terlebih dahulu mengontak *root server*. Root server ini ialah sekumpulan Server DNS yang berada pada hirarki tertinggi. Root server tentu saja tidak memiliki informasi IP address host **students.itb.ac.id** ini. Ia hanya akan memberikan referensi, bahwa untuk mencari nama host

dengan domain ***id***, hendaknya server kita menghubungi name server yang *authoritative* untuk domain ***id*** (indonesia) ini. Ketika name server ***id*** dihubungi, dia pun akan melakukan hal yang sama. Dia hanya akan menjawab dengan memberikan nama name server yang bertanggung jawab atas domain **ac.id**. Pertanyaan ke Name Server untuk domain **ac.id** ditanya akhirnya menghasilkan referensi bahwa seharusnya pertanyaan ini diberikan ke name server untuk domain **itb.ac.id**. Name server inilah yang akhirnya memberikan jawaban sebenarnya atas pertanyaan di atas.

Dalam mencari jawaban atas *query* (pertanyaan) dari resolver di atas, Name Server melakukan proses iteratif (*iterative query*). Dalam proses iteratif ini, name server kita menelusuri name server lain satu per satu, berdasarkan referensi yang diberikan oleh masing-masing name server lain tersebut.

Sebaliknya resolver pada ilustrasi di atas melakukan query yang bersifat rekursif (*recursive query*) pada name server. Resolver memaksa Name Server untuk mencari seluruh jawaban, untuk kemudian memberikan hasilnya kepadanya. Resolver menolak untuk menerima jawaban yang berupa referensi name server lain, yang seharusnya berhak untuk ditanyai.

Resolver dapat pula meng-query name server secara iteratif. Jika hal ini dilakukan, name server hanya akan melihat database lokal atau cache miliknya. Jika data yang dimaksud tidak ada, name server akan "menyerah". Jika data tersebut terdapat di database lokalnya, dia akan memberikan jawabannya. Jika data tersebut ada di cache filenya, dia akan menjawab sambil memberitahukan bahwa jawaban ini bersifat

non authoritative (bukan berasal dari server yang authoritative, melainkan dari cache).

Waktu yang diperlukan bagi proses resolusi di atas relatif lama, karena berkaitan dengan proses query ke *name server authoritative* (yang bisa jadi berada di tempat yang jauh). Untuk mempercepat proses ini, dilakukan satu mekanisme yang dikenal sebagai *caching*. Ketika suatu name-server melakukan proses *resolution* dari satu referensi ke referensi yang lain hingga ditemukan jawabannya, ia "mempelajari" korelasi antara alamat-alamat server yang pernah dihubunginya. Alamat ini disimpan (dalam batas waktu tertentu – Time to Live TTL) untuk kemudian digunakan kembali jika diperlukan. Dengan mengingat data tersebut, Name Server dapat mempercepat proses untuk query berurut, baik pada domain-name yang sama atau domain-name yang berkorelasi dengan sebelumnya.

Misalnya name-server kita pernah melakukan resolusi address **eecs.berkeley.edu**, dan dalam proses tersebut name-server meng-cache address name-server **eecs. berkeley.edu** dan **berkeley edu**. Sekarang jika kita meng-query untuk **baobab.cs.berkeley.edu**, name-server kita akan mem-bypass query ke root name server, karena ia telah mengenali bahwa **berkeley.edu** yang telah ia ketahui berkorelasi penuh dengan data yang diminta. Name-server ini akan memulai resolution dengan query pada name-server **berkeley.edu**. Dengan cara ini waktu untuk resolution telah direduksi dan aplikasi dapat berjalan lebih cepat.

4.4. Mengkonfigurasi Domain Name Server

Setelah mengetahui cara kerja DNS secara umum, kita akan masuk ke proses konfigurasi DNS. Dalam sistem operasi UNIX, software implementasi DNS yang paling sering di gunakan ialah BIND.

4.4.1. BIND

BIND (*Berkeley Internet Name Domain*) adalah software implementasi DNS yang dibuat oleh **Kevin Dunlap** untuk sistem operasi Unix Berkeley 4.3BSD. Hingga saat ini, BIND telah di-*port* ke berbagai sistem operasi Unix lain, dan telah menjadi bagian standard yang ditawarkan oleh vendor-vendor Unix. Walaupun demikian BIND bukan yang pertama, karena sebelumnya **Paul Mockapetris** terlebih dulu menulis JEEVES yang menjadi implementasi pertama spesifikasi DNS. Dalam pembahasan selanjutnya, kita akan menggunakan terminologi BIND.

Jika pada server Anda belum tersedia software ini, Anda dapat mengambilnya melalui Internet, di URL **ftp://ftp.isc.org/isc/bind/** serta meng-compile-nya sesuai instruksi yang terdapat pada file tersebut.

4.4.2. Mengkonfigurasi BIND

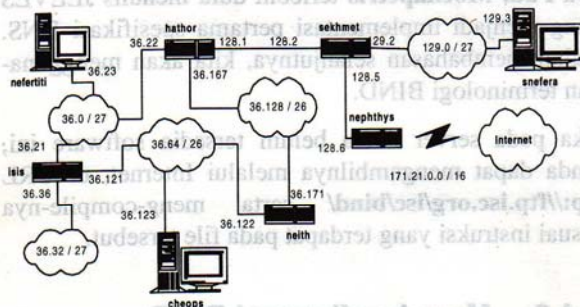
Untuk menjalankan Server DNS pada suatu host, diperlukan file-file berikut:

```
/etc/named
/etc/named.boot
zone file
```

named (dibaca name-di) merupakan file *executable*. Sedangkan **named.boot** dan **zone file** merupakan file text.

Named.boot merupakan file yang pertama kali dibaca oleh ***named*** saat ia dijalankan. File ini berisi informasi **inisialisasi domain,** yaitu tipe Server DNS yang dijalankan ini, daftar zone tempat Server DNS ini memiliki *authority*, serta lokasi file atau server lain tempat Server DNS ini bisa mendapatkan data awalnya. Sedangkan zona file berisi data *resource record* untuk masing-masing host. Arti kata *resource record* akan diterangkan pada subbab berikutnya.

Konfigurasi Jaringan

Dalam menerangkan konfigurasi *boot script* dan *zona file* ini, akan digunakan contoh konfigurasi jaringan seperti berikut.

Gambar 4.6 Network untuk konfigurasi DNS

Seluruh jaringan pada Gambar 4.6 berada di bawah domain alexandria.cnrg.net. Jaringan di atas terdiri atas lima buah router (*isis*, *hathor*, *neith*, *sekhmet*,

nephthys), dua buah Server DNS (*cheops* dan *snefera*), dan sebuah mail server (*nefertiti*) .

Seluruh Penamaan ini diambil berdasarkan nama nama dewa, dewi, ratu, raja, serta kota-kota di jaman Mesir Kuno. Informasi yang cukup lengkap tentang kebudayaan Mesir Kuno ini dapat Anda diperoleh melalui http://160.227.122.58/egypt/. Sementara itu nama CNRG merupakan singkatan dari *Computer Network Research Group*, nama lembaga riset kami di ITB

Konfigurasi boot script Server DNS

named.boot

Pada subbab sebelumnya telah diterangkan tentang named.boot. File ini harus berisi informasi inisialisasi Server DNS yang bersangkutan. Dalam file ini terdapat beberapa perintah, seperti pada Tabel 4.2 berikut:

Tabel 4.2 Perintah pada named.boot

Directory	Mendefenisikan directory tempat
Primary	Mendeklarasikan Server DNS
Secondary	Mendeklarasikan Server DNS
Cache	Mendefenisikan cache file
Forwarders	Mendefenisikan daftar server untuk
Slave	Memfungsikan Server DNS hanya

Isi named.boot akan berkaitan dengan konfigurasi DNS yang kita buat, sebagaimana terlihat pada gambar di atas. Karenanya, ada baiknya kita definisikan dengan lebih rinci kondisi jaringan yang kita miliki tersebut:

Jaringan pada gambar di atas memiliki range IP Address 132.92.XXX.XXX dengan masing-masing IP Address tertera pada gambar di atas.

Jaringan di atas ada di bawah domain name **alexandria cnrg.netE**

Secara topologi jaringan di atas dapat dilihat sebagai dua jaringan/network yang dihubungkan dengan interface serial. Kedua network tersebut adalah network 132.92.128.0/27 dan network sisanya. Karenanya, agar pada jaringan terjadi distribusi beban, diperlukan dua buah Server DNS untuk melayani masing-masing bagian jaringan. Name server tersebut ialah *snefera. alexandria.cnrg.net* (132.92.129.3) dan *cheops. alexandria.cnrg.net* (132.92.36.123)

Karena hanya akan ada satu primary server untuk domain *alexandria.cnrg.net.*, maka kita tetapkan *snefera.alexandria.cnrg.net.* sebagai primary name server dan *cheops.alexandria.cnrg.net.* sebagai secondary server.

Berdasarkan informasi di atas kita dapat membuat file `named.boot` yang dibutuhkan. Mula-mula kita buat dulu `named.boot` untuk Server DNS primary, yaitu ***snefera.alexandria.cnrg.net***.

Isi filenya adalah sebagai berikut:

```
;
; boot file for primary server
; snefera.alexandria.cnrg.net
;
directory                               /etc/named.data
cache   .                               db.cache
primary  0.0.127.IN-ADDR.ARPA           db.local
; domain source host/file               backup file
primary alexandria.cnrg.net.            db.alexandria
```

Keterangan:

Komentar di awali dengan karakter ';'

Pada baris kelima direktori /etc/named.data digunakan untuk menyimpan zona file. Direktori ini harus telah ada sebelumnya dan seluruh zona file akan disimpan di direktori /etc/named.data.

Pada baris keenam, server DNS menjalankan proses *caching* data DNS dan file yang digunakan untuk inisialisasi cache, yaitu /etc/named.data/db.cache.

Baris ketujuh adalah deklarasi bahwa DNS merupakan primary server untuk loopback domain. Zona file untuk loopback domain adalah /etc/named.data/db.local. Zona file boleh diberi nama sesuai dengan keinginan administrator jaringan. Tapi yang penting bahwa nama tersebut mewakili zona tertentu

Baris kesembilan berarti bahwa server DNS ini menjadi primary name server untuk domain *alexandria.cnrg.net.*, dan zona filenya adalah /etc/named.data/db.alexandria. zona file ini harus dibuat terlebih dahulu. Proses pembuatan zona file ini akan diterangkan kemudian.

Sedangkan boot file untuk host *cheops. alexandria.cnrg.net.* adalah sebagai berikut

```
;
; boot file for secondary server
; cheops.alexandria.cnrg.net
;
directory                            /etc/named.data
cache      .                         db.cache
primary 0.0.127.IN-ADDR.ARPA         db.local
; domain source host/file            backup file
secondary alexandria.cnrg.net. 132.92.129.3 db.alexandria.bak
```

Perbedaan isi file named.boot ini dengan named.boot sebelumnya terdapat pada baris ke delapan. Baris ini berarti bahwa server ini menjadi secondary name server bagi domain alexandria.cnrg.net, primary server adalah host dengan IP address 132.92.129.3 (yaitu *snefera. alexandria.cnrg.net*) sedangkan zona file (hasil zona transfer) terletak di file /etc/named.data/db.alexandria. bak.

Konfigurasi reverse domain pada named.boot

Di samping pemetaan dari hostname ke IP address, dalam jaringan TCP/IP diperlukan juga pemetaan dari IP address ke hostname. Pemetaan ini merupakan pemetaan balik dari pemetaan hostname ke IP address. Proses ini disebut *reverse mapping* (pemetaan balik), sementara domainnya disebut *reverse domain.*. Reverse domain biasanya diperlukan untuk pembuatan statistik akses ke suatu server (mengetahui host dari domain mana saja yang mengakses server tertentu, untuk kemudian disimpan dalam log file). Di samping itu reverse domain juga diperlukan untuk security jaringan (*authorization check*). Bila menggunakan host table (/etc/hosts, HOSTS.TXT) maka pemetaan nama host ke IP address merupakan pemetaan satu ke satu. Resolver akan mencari hostname pada host tabel secara berurutan (sekuensial).

Dengan DNS proses pencarian IP address dari suatu nama host dapat dengan mudah dilakukan. Tapi proses pencarian nama host dari suatu host dengan IP address tertentu memerlukan proses pencarian yang cukup lama, karena harus dilacak ke seluruh domain name server. Solusi yang digunakan adalah dengan membuat suatu domain dengan menggunakan IP address sebagai

domain. Pada jaringan TCP/IP top level domain yang menggunakan IP address sebagai domain diberi nama **in-addr.arpa**. Pemberian nama subdomain di bawah top level domain ini mengikuti aturan sebagai berikut:

- Subdomain dibentuk dengan menuliskan subdomain dalam format representasi IP address dalam bentuk *dot-octet*.
- Pembentukan subdomain di bawah top level domain dimulai dari oktet pertama dari IP address (IP address terdiri atas 32 bit=4 oktet) dan subdomain selanjutnya dibentuk dari oktet kedua dan demikian seterusnya.

Contoh:

Sebuah network dengan IP address 132.92.XXX.XXX (Network Klas B, XXX = variable dari 0 s.d 255) dikoordinir oleh Server DNS *snefera.alexandria. cnrg.net.* Agar Server DNS ini dapat menjadi server untuk reverse domain pada IP address di atas, maka reverse domain yang harus dibuat adalah

92.132.in-addr.arpa

Network dengan IP address 132.92.XXX.XXX bila direpresentasikan dalam bentuk *dot-octet* adalah 132.92.

Oktet pertama dari IP address network di atas adalah 132, oktet kedua 92, maka subdomain di bawah top level domain in-addr.arpa adalah **132.in-addr.arpa.** Subdomain berikutnya adalah oktet kedua yaitu 92, maka di bawah sub-domain 167.in-addr.arp terdapat lagi subdomain **92.132.in-addr.arpa** .

Apabila IP address 132.92.XXX.XXX didelegasikan ke beberapa organisasi dan tiap organisasi mempunyai DNS sendiri maka setelah sub-domain **92.132.in-addr.arpa** bisa juga dibentuk beberapa subdomain lagi seperti **1.92.132..in-addr.arpa, 2.92.132.in-addr.arpa, 3.92.132.in-addr.arpa** dan seterusnya

Agar Server DNS *snefera.alexandria.cnrg.net.* dapat menjadi primary server untuk reverse domain **92.132.in-addr.arpa**, pada file named.boot harus ditambahkan baris berikut:

```
; primary reverse domain server
primary 92.132.IN-ADDR.ARPA db.132.92
```

Agar Server DNS *cheops.alexandria.cnrg.net.* menjadi secondary server untuk reverse domain **92.132.in-addr.arpa**, pada named.bootnya harus ditambahkan baris berikut:

```
; secondary reverse domain server
secondary  92.132.IN-ADDR.ARPA  132.92.129.3 db.132.92.bak
```

Arti dari dua baris di atas: 132.92.129.3 ialah IP address dari primary server (*snefera.alexandria. cnrg.net.*), sedangkan zona file untuk *reverse domain* ini terletak di /etc/namedb/db.132.92.bak.

Selain daftar file reverse address untuk host, dalam named.boot perlu juga dicantumkan nama file yang berisi PTR record/reverse address untuk loopback ineterface (IP 127.0.0.1). untuk itu pada named.boot ditambahkan baris berikut

```
Primary 0.0.127.IN-ADDR.ARPA                db.127.0.0
```

Zona file tempat penyimpanan PTR record loopback interface sesuai baris di atas ialah `db.127.0.0`

db.cache

Cache file adalah file yang digunakan pada saat menjalankan Caching-only server. File ini digunakan untuk memberikan informasi kea Server DNS tentang root server yang harus dihubungi untuk mengetahui informasi host-host yang tidak terdapat pada domain lokal. Cache file ini sangat penting apabila jaringan kita telah terhubung ke Internet. Karena untuk mendapatkan IP address hostname di Internet tidak mungkin melakukan zona transfer antar seluruh Server DNS.

Format dari file db.cache ialah sebagai berikut.

```
; This file holds the information on root name servers needed to
;       initialize cache of Internet domain name servers
;       (e.g. reference this file in the "cache  .  <file>"
;       configuration file of BIND domain name servers).
;
;       This file is made available by InterNIC registration services
;       under anonymous FTP as
;           file                /domain/named.root
;           on server           FTP.RS.INTERNIC.NET
;       -OR- under Gopher at    RS.INTERNIC.NET
;           under menu          InterNIC Registration Services (NSI)
;              submenu          InterNIC Registration Archives
;           file                named.root
;
;       last update:    Aug 22, 1997
;       related version of root zone:   1997082200
;
;
; formerly NS.INTERNIC.NET
;
.                        3600000  IN  NS    A.ROOT-SERVERS.NET.
A.ROOT-SERVERS.NET.      3600000      A     198.41.0.4
;
; formerly NS1.ISI.EDU
;
.                        3600000      NS    B.ROOT-SERVERS.NET.
B.ROOT-SERVERS.NET.      3600000      A     128.9.0.107
```

```
;
; formerly C.PSI.NET
;
.                    3600000    NS   C.ROOT-SERVERS.NET.
C.ROOT-SERVERS.NET.    3600000    A    192.33.4.12
;
; formerly TERP.UMD.EDU
;
.                    3600000    NS   D.ROOT-SERVERS.NET.
D.ROOT-SERVERS.NET.    3600000    A    128.8.10.90
;
; formerly NS.NASA.GOV
;
.                    3600000    NS   E.ROOT-SERVERS.NET.
E.ROOT-SERVERS.NET.    3600000    A    192.203.230.10
;
; formerly NS.ISC.ORG
;
.                    3600000    NS   F.ROOT-SERVERS.NET.
F.ROOT-SERVERS.NET.    3600000    A    192.5.5.241
;
; formerly NS.NIC.DDN.MIL
;
.                    3600000    NS   G.ROOT-SERVERS.NET.
G.ROOT-SERVERS.NET.    3600000    A    192.112.36.4
;
; formerly AOS.ARL.ARMY.MIL
;
.                    3600000    NS   H.ROOT-SERVERS.NET.
H.ROOT-SERVERS.NET.    3600000    A    128.63.2.53
;
; formerly NIC.NORDU.NET
;
.                    3600000    NS   I.ROOT-SERVERS.NET.
I.ROOT-SERVERS.NET.    3600000    A    192.36.148.17
;
; temporarily housed at NSI (InterNIC)
;
.                    3600000    NS   J.ROOT-SERVERS.NET.
J.ROOT-SERVERS.NET.    3600000    A    198.41.0.10
;
```

```
; housed in LINX, operated by RIPE NCC
;
.                 3600000    NS   K.ROOT-SERVERS.NET.
K.ROOT-SERVERS.NET.     3600000    A    193.0.14.129
;
; temporarily housed at ISI (IANA)
;
.                 3600000    NS   L.ROOT-SERVERS.NET.
L.ROOT-SERVERS.NET.     3600000    A    198.32.64.12
;
; housed in Japan, operated by WIDE
;
.                 3600000    NS   M.ROOT-SERVERS.NET.
M.ROOT-SERVERS.NET.     3600000    A    202.12.27.33
; End of File
```

Untuk mendapatan file ini, Anda harus melakukan ftp ke **rs.internic.net**, dengan lokasi direktori /domain/named.root. ubah nama file ini menjadi db.cache dan letakkan di bawah direktori /etc/namedb

Konfigurasi Zona File

Konsep Resource Record

Zona file merupakan file yang berisi informasi yang berkaitan dengan suatu domain. Dalam zona file, tersedia satu set informasi untuk setiap domain. Set informasi untuk satu domain bisa berupa IP address, nama alias dari domain/host tersebut, nama Name Server yang bertanggung jawab untuk domain tersebut dan sebagainya. Masing-masing informasi dalam kumpulan informasi ini disebut sebagai *resource record*. Sedangkan standard penulisannya disebut sebagai *standard resource record*.

Spesifikasi *standard resource record* ini terdapat dalam RFC1035. Format standard dari *resource record* ialah sebagai berikut:

{name} {ttl} addr-class Record Type Record Specific data

Field pertama ialah nama dari domain record. Penulisan field ini harus selalu dimulai dari kolom pertama. Untuk semua *resource record* selain *resource record* pertama dalam satu file, field ini boleh dikosongkan. Jika hal ini dilakukan, maka domain record-nya mengikuti domain yang pertama kali disebut.

Field kedua berisi *time to live*, yaitu waktu lamanya data ini boleh disimpan dalam database. Jika filed ini dikosongkan, maka *time to live* (TTL) yang dipakai ialah TTL yang disebutkan dalam *resource record* Source of Authority (SOA), yang akan diterangkan setelah ini.

Field ketiga, berisi address class. Hanya satu class yang disupport, yaitu IN, untuk Internet address. Field keempat berisi data tipe *resource record*, dan isi field berikutnya tergantung pada jenis *resource record* yang bersangkutan. Seluruh data ini digunakan oleh name server secara *case insensitive*, yaitu tidak memperdulikan perbedaan huruf besar dan kecil.

Resource record yang digunakan dalam DNS ialah sebagai berikut:

***Start of Authority Record* (SOA)**

Fungsi Start of Authority Record (SOA) adalah mendefenisikan hostname yang merupakan awal dari suatu zone. Untuk setiap zone hanya mempunyai

sebuah SOA. SOA biasanya dideklarasikan pada awal zona file.

Format:

```
[zone] IN SOA origin contact (
serial
refresh
retry
expire
minimum
)
```

Komponen SOA record terdiri atas:

zone Komponen ini mendefinisikan nama dari zona. SOA record terdiri atas zone yang diawali dengan karakter at-sing ('@'). Dengan penulisan ini berarti domain yang dideklarasikan pada boot script yang diawali dengan statement "*primary*" merupakan asal dari zone tersebut.

origin Mendeklarasikan hostname yang merupakan primary master server untuk domain. Hostname biasanya ditulis secara FQDN, misalnya **snefera.alexandria.cnrg.net.**

contact Mendeklarasikan e-mail address administrator yang bertanggung-jawab terhadap domain. Standard penulisan e-mail administrator adalah **user.hostname**, misalnya **cnrg. snefera.alexandria.cnrg.net.** Administrator domain adalah user dengan nama **cnrg** pada host **snefera.alexandria.cnrg.net**. Jika administrator memiliki alamat email dengan tanda titik "." (misalnya **datuk.maringgih**

@snefera.alexandria.cnrg.net.), penulisannya contactnya adalah sebagai berikut: **datuk\.maringgih.snefera.alexandria.cnrg.net.**

serial Merupakan nomor seri dari zona file. ***Serial number ini harus bertambah setiap ada perubahan data pada zona file***. Serial number ini digunakan oleh *secondary server* untuk melakukan pemeriksaan ada tidaknya perubahan *zona file* pada *primary server*. Untuk melakukan pengecekan, *secondary server* akan melihat serial number. Apabila serial number di *primary server* lebih besar dari serial number yang terdapat pada zona file di *secondary server*, maka *secondary server* akan melakukan *full zona transfer* dari *primary server*. Apabila tidak ada perubahan serial number maka secondary server berasumsi bahwa tidak perubahan zona file pada primary server.

Sebaiknya format serial number yang digunakan adalah: YYMMddhhmm (YY=dua angka terakhir dari tahun, MM=bulan, dd=tanggal, hh=jam, mm=menit). Artinya bahwa setiap ada perubahan pada zona file maka serial number diganti dengan angka-angka tahun, bulan, tanggal, jam dan menit tanggal perubahan. Contoh: jika perubahan dilakukan tanggal 2 Bulan Maret Tahun 1998 jam 11.35 WIB, maka serial number yang digunakan ialah 9803021135. Dengan cara ini maka setiap ada perubahan maka serial number pasti bertambah besar.

refresh Komponen ini mendeklarasikan selang waktu (dalam detik) yang diperlukan oleh secondary server untuk memeriksa perubahan zona file pada primary server. Pada setiap selang waktu yang telah ditentukan, secondary server akan memeriksa terhadap serial number untuk mengetahui apakah ada perubahan zona file. Selang waktu ini dipilih berdasarkan dinamika perubahan zona file antar Server DNS. Biasanya perubahan zona file hanya bersifat harian, maka sebaiknya selang waktu dapat dipilih satu hari (24jam x 3600 detik).

retry Komponen ini menentukan berapa lama (dalam detik) secondary server menunggu untuk mengulang pengecekan terhadap primary server apabila primary server tidak memberikan respon pada saat proses refresh. Jangan menggunakan nilai *retry* yang terlalu kecil karena pengulangan dalam waktu singkat tidak menghasilkan apa-apa karena ada kemungkinan primary server sedang down. Sebaiknya gunakan *retry* sekitar satu jam lebih.

expire Komponen ini menentukan berapa lama (dalam detik) zona file dipertahankan pada secondary server apabila secondary server tidak dapat melakukan zona refresh. Apabila setelah masa expire, secondary server tidak dapat melakukan zona refresh maka secondary server akan menghapus file tersebut dari zona file. Sebaiknya nilai komponen ini cukup besar (lebih besar dari 30 hari) dan

untuk link yang kurang reliable sebaiknya sekitar enam bulan ataupun satu tahun.

minimum Komponen ini menentukan nilai default *time to live* (TTL) untuk semua *resource record* pada zona file. Sebaiknya nilai ini dibuat sebesar mungkin, karena jarang sekali perubahan pada hostname begitu hostname tersebut diberi IP address dan MX record.

Berikut ini ialah contoh penulisan SOA didalam zona file db.alexandria, yang berada di Server DNS **snefera.alexandria.cnrg.net**

```
; domain alexandria.cnrg.net
@ IN    SOA                          snefera.alexandria.cnrg.net.
cnrg.snefera.alexandria.cnrg.net. (
        9803021135                   ; Serial
        10800                        ; Refresh every 3 hours
        3600                         ; Retry every hour
        6048000                      ; Expire after 10 week
        8640000 )                    ; ttl of 100 day
```

Name Server Record **(NS)**

NS record merupakan identifikasi authoritative server untuk suatu zona. *Authoritative server* untuk suatu zona sebaiknya lebih dari satu sebagai tindakan preventif apabila primary master server tidak bisa diakses oleh secondary server.

Format:

[*domain*] **IN NS** *server*

Komponen Name Server Record adalah sebagai berikut:

domain Authoritative server untuk domain ini adalah Server DNS yang tertulis pada komponen server.

server Hostname dari komputer yang merupakan authoritative Server DNS untuk domain yang tercantum pada komponen domain. Komponen ini ditulis secara FQDN.

Berikut ini adalah contoh penulisan Standard Resource Record SOA dan NS untuk domain alexandria.cnrg.net.

```
; domain alexandria.cnrg.net
@      IN     SOA                         snefera.alexandria.cnrg.net.
cnrg.snefera.alexandria.cnrg.net. (
            9803021135                    ; Serial
            10800                         ; Refresh every 3 hours
            3600                          ; Retry every hour
            6048000                       ; Expire after 10 week
            8640000 )                     ; ttl of 100 day
            IN   NS                       snefera.alexandria.cnrg.net.
            IN   NS                       cheops.alexandria.cnrg.net.
```

***Address Record* (A)**

Fungsi Address Record (A) adalah untuk memetakan hostname ke IP address.

Format:

[*host*] IN A *address*

Komponen Address Record adalah sebagai berikut:

host Nama host yang hostnya seperti yang tercantum pada komponen *address*, hostname ditulis relatif terhadap domain dari host tersebut. Misalkan address record dari *neith.alexandria.cnrg.net.* akan dituliskan

pada zona file db.alexandria maka yang dituliskan pada zona file hanya **neith.**

address adalah IP address untuk host dan ditulis dalam bentuk *dotted-decimal*.

Jika suatu host yang bersifat *multihoming*, yaitu host yang terhubung ke beberapa network dengan menggunakan lebih dari satu network interface maka record address host tersebut dapat lebih dari satu.

Berikut ini adalah daftar hostname dan IP address dari router dan Server DNS sesuai dengan Gambar 4.6.

Tabel 4.3 Nama dan IP address tiap host

Hostname	IP Address
Nephthys	132.92.128.6
	132.92.128.65
Sekhmet	132.92.128.2
	132.92.128.33
	132.92.129.2
Hathor	132.92.36.22
	132.92.36.167
	132.92.128.1
Neith	132.92.36.122
	132.92.36.71
	132.92.128.30
Isis	132.92.36.21
	132.92.36.36
	132.92.36.121
Snefera	132.92.129.3
Cheops	132.92.36.123
Nefertiti	132.92.36.23

Berdasarkan Tabel 4.3, berikut ini diberikan contoh penulisan Standard Resource Record SOA, NS dan A untuk domain *alexandria.cnrg.net.*

```
; domain alexandria.cnrg.net
@ IN   SOA                    snefera.alexandria.cnrg.net.
cnrg.snefera.alexandria.cnrg.net. (
     9803021135               ; Serial
     10800                    ; Refresh every 3 hours
     3600                     ; Retry every hour
     6048000                  ; Expire after 10 week
     8640000 )                ; ttl of 100 day
     IN  NS                   snefera.alexandria.cnrg.net.
     IN  NS                   cheops.alexandria.cnrg.net.
Snefera IN A 132.92.129.3
Cheops                        IN A 132.92.36.123
Nephthys                      IN A 132.92.128.6
                              IN A 132.92.128.65
Sekhmet                       IN A 132.92.128.2
                              IN A 132.92.128.5
                              IN A 132.92.129.2
Hathor                        IN A 132.92.36.22
                              IN A 132.92.36.167
                              IN A 132.92.128.1
Neith                         IN A 132.92.36.122
                              IN A 132.92.36.71
                              IN A 132.92.128.30
Isis                          IN A 132.92.36.21
                              IN A 132.92.36.36
                              IN A 132.92.36.121
Nefertiti                     IN A 132.92.36.23
```

Zona file kedua yang dibuat ialah db.local. Zona file ini tercantum dalam file named.boot, dan berisi data IP localhost melalui loopback interface (127.0.0.1). File ini diletakkan pada direktori /etc/named.data/

Contoh dari `db.local` adalah sebagai berikut:

```
@     IN     SOA    snefera.alexandria.cnrg.net.
cnrg.snefera.alexandria.cnrg.net. (
```

```
        9803021135  ; serial
        86400       ; refresh every 1 day
        3600        ; retry every 1 hour
        2592000     ; expire afer 1 month eq. 30 day
        25200  )    ; min ttl 7 hour
   IN NS snefera.alexandria.cnrg.net
   localhost    IN    A    127.0.0.1
```

***Mail Exchanger Record* (MX)**

MX record digunakan untuk mengarahkan mail untuk suatu host ataupun suatu domain ke host yang berfungsi sebagai mail server. MX record sangat berguna untuk suatu domain yang tidak menjalankan mail software. Mail yang ditujukan untuk domain ini akan diarahkan ke host yang menjalankan mail software.

Format:

[*name*] IN MX *preference host*

Komponen Mail Exchanger Record adalah sebagai berikut:

name — Hostname ataupun domain tujuan pengiriman mail. Bila tujuan pengiriman adalah suatu domain pada suatu zona file, maka bagian ini cukup dikosongkan.

preference — Menentukan tingkat prioritas mail exchanger yang akan digunakan untuk me-redirect mail ke [*name*]. Sebuah host ataupun suatu domain dapat memiliki beberapa mail exchanger. Mail exchanger yang digunakan pertama kali adalah mail exchanger dengan prioritas tertinggi dan apabila mail exchanger ini gagal dihubungi maka digunakan prioritas berikutnya dan demikian seterusnya.

	Mail exchanger dengan preference terendah merupakan mail exchanger dengan prioritas tertinggi. Apabila suatu host tidak mempunyai MX record maka remote host akan berusaha mengirimkan mail langsung ke host tujuan. Hal ini tidak disarankan, karena ada kemungkinan suatu host tidak dapat diakses karena link terputus ataupun sedang dalam kondisi perawatan. Sebaiknya setiap host mempunyai MX Record
host	Adalah hostname dari mail exchanger yang digunakan untuk mengarahkan mail ke host ataupun domain yang didefenisikan pada field *name*.

Berikut ini adalah contoh suatu zona file yang mempunyai MX record untuk domain dan MX record untuk host.

```
; domain alexandria.cnrg.net
@ IN  SOA                          snefera.alexandria.cnrg.net.
cnrg.snefera.alexandria.cnrg.net. (
     9803021135                    ; Serial
     10800                         ; Refresh every 3 hours
     3600                          ; Retry every hour
     6048000                       ; Expire after 10 week
     8640000 )                     ; ttl of 100 day
     IN  NS                        snefera.alexandria.cnrg.net.
     IN  NS                        cheops.alexandria.cnrg.net.
     IN  MX 40                     snefera.alexandria.cnrg.net.
Snefera IN A 132.92.129.3
Cheops                             IN A 132.92.36.123
Nefertiti                          IN A 132.92.36.23
                                   IN MX 40 Snefera
                                   IN MX 30 Cheops
                                   IN MX 10 Nefertiti
Nephthys                           IN A 132.92.128.6
```

```
                            IN A 132.92.128.65
Sekhmet                     IN A 132.92.128.2
                            IN A 132.92.128.5
                            IN A 132.92.129.2
Hathor                      IN A 132.92.36.22
                            IN A 132.92.36.167
                            IN A 132.92.128.1
Neith                       IN A 132.92.36.122
                            IN A 132.92.36.71
                            IN A 132.92.128.30
Isis                        IN A 132.92.36.21
                            IN A 132.92.36.36
                            IN A 132.92.36.121
```

Pada contoh di atas, tertulis MX record sebagai berikut:

```
nefertiti                IN A        132.92.36.23
IN MX  40 snefera.alexandria.cnrg.net.
IN MX  30 cheops.alexandria.cnrg.net.
IN MX  10                nefertiti
```

MX record di atas dibuat dengan memberikan prioritas tertinggi ke host yang bersangkutan (*nefertiti*). Jika pengiriman mail ke *nefertiti* gagal, mail dicoba dikirimkan ke *cheops* yang merupakan *mail exchanger*-nya. Jika pengiriman ke *cheops* juga gagal, dilakukan pengiriman mail ke mail exchanger yang kedua, yaitu *snefera*.

Selang pemilihan preference MX Record biasanya dibuat berselisih sepuluh angka. Selisih ini dibuat sedemikian rupa agar apabila ada penambahan mail server, dapat dilakukan penyisipan pada MX record yang telah ada sebelumnya.

Selain MX record untuk nefertiti, pada file di atas juga terdapat baris berikut:

IN MX 40 snefera.alexandria.cnrg.net

Berdasarkan baris di atas, host **snefera.alexandria. cnrg.net** menjadi *mail exchanger* untuk domain *alexandria.cnrg.net*. E-mail yang diarahkan ke **user@alexandria.cnrg.net** akan sampai ke **user@snefera.alexandria.cnrg.net**. Hal Ini dilakukan untuk mempermudah pengiriman e-mail, sehingga pengirim email tidak harus mengetahui secara tepat hostname tempat user berada. Contoh yang lain, misalkan ada user yang akan mengirimkan mail ke President Direktur Intel Corp., maka pengirim tersebut dapat mengirimkan mail yang ditujukan kepada **director@intel.com.** Apabila pada zona file terdapat MX record untuk domain **intel.com,** maka mail tersebut akan dikirimkan ke mail server untuk domain tersebut. Mail server akan mendistribusikan mail tersebut ke tujuan yang sebenarnya.

Canonical Name

Fungsi Canonical Name adalah mendefinisikan alias name atau nickname untuk suatu host.

Format:

[nickname] **IN CNAME** host

Komponen CNAME record adalah sebagai berikut:

nickname Adalah alias name untuk host yang tercantum pada host.

host Hostname yang alias name-nya tercantum pada nickname. Hostname harus ditulis secara FQDN dan tidak dianjurkan berupa alias name (karena mengakibatkan looping).

Resource record Canonical Name berfungsi mendefinisikan nama alias suatu host. Satu alias hanya boleh berlaku untuk satu nama host. Seluruh resource record yang lain hanya boleh diasosiasikan dengan nama host yang asli, dan bukan dengan nama alias. Misalkan kita ingin membuat mail.alexandria.cnrg.net. sebagai nama alias dari nefertiti, pada zona file dituliskan satu baris berikut:

```
mail     IN CNAME          nefertiti.alexandria.cnrg.net.
```

Pembuatan nama alias untuk suatu host ini berguna untuk mempermudah mengingat nama host, terutama jika host yang asli mengalami perubahan. Host dengan nama **www.cnrg.net** lebih mudah diingat dibandingkan dengan snefera.alexandria.cnrg.net. Jika kita ingin snefera ini menjadi web server, maka host ini dapat kita beri nama alias **www.cnrg.net**. Jika pada suatu saat kita ingin memindahkan web tersebut ke mesin lain, kita cukup mengaliaskan nama server tersebut dengan nama yang sudah dikenal orang sebelumnya. Dengan demikian, orang yang menggunakan nama lama masih tetap bisa mengakses host tujuan.

Konfigurasi Zona File untuk reverse domain

Untuk pembuatan data reverse-domain, digunakan resource record berupa PTR (pointer) record. PTR record ini akan memetakan IP address ke domain-name.

PTR Record
Fungsi PTR Record adalah mendefinisikan reverse address untuk suatu *hostname*.

Format:

[octet] **IN PTR** hostname

Komponen PTR record adalah sebagai berikut:

octet Adalah octet terakhir dari IP address pada *hostname* yang bersangkutan.

host Hostname dari IP address yang bersangkutan. Hostname ini harus ditulis secara FQDN.

Berikut ini ditunjukkan contoh pemakaian PTR record pada file **db.132.92.128** . file ini berisi reverse address bagi hostname dengan IP Address 132.92.128.XXX

```
; contoh PTR record
@ IN SOA dns.paume.ITB.ac.id. cnrg.ITB.ac.id. (
9803021135                                  ;serial
86400       ;refresh every 12 hours
10800       ;retry every 3 hours
6048000                                     ;expire after 10 week
8640000                                     ;minimum TTL of 100 day
)
        IN NS snefera.alexandria.cnrg.net.
        IN NS cheops.alexandria.cnrg.net.
;
1       IN PTR          hathor.alexandria.cnrg.net.
2       IN PTR          sekhmet.alexandria.cnrg.net.
5       IN PTR          sekhmet.alexandria.cnrg.net.
6       IN PTR          nephthys.alexandria.cnrg.net.
30      IN PTR          neith.alexandria.cnrg.net.
;
```

PTR record lain yang dibutuhkan oleh Server DNS ialah PTR record untuk loopback interface. Record ini diletakkan dalam file `db.127.0.0` sebagaimana yang tercantum dalam file named.boot. bentuk file ini adalah sebagai berikut:

```
@     IN    SOA    snefera.alexandria.cnrg.net.
cnrg.snefera.alexandria.cnrg.net. (
    9803021135    ;Serial
    86400  ;refresh rate every 24 hours that's equal 1 day
```

```
     3600           ;retry every one hour
     2592000        ;expire after 1 month
     25200 )        ;ttl is set at seven hours
      IN    NS      snefera.alexandria.cnrg.net.
1     IN    PTR     localhost.
```

4.5. Menjalankan Server DNS

Setelah menyiapkan file konfigurasi (**/etc/named.boot**) dan zone file, maka tahap selanjutnya adalah menjalankan Server DNS tersebut. Implementasi BIND menggunakan name-server daemon yand disebut **named** (dibaca *name-d*). Perintah untuk menjalankan daemon tersebut adalah sebagai berikut.

named [**-d** *level*] [**-p** *port*] [**-b** *bootfile*]

-d *level* Digunakan untuk menentukan level penyimpanan informasi log debugging dalam file **/usr/tmp/named.run**. Argumen dari level adalah bilangan dari 1 hingga 9. Semakin tinggi level penyimpanan, maka semakin detil informasi yang disimpan dalam file **/usr/tmp/name.run** dan file tersebut akan membesar dalam waktu singkat

-p *port* Menentukan port UDP/TCP yang digunakan oleh **named**. Nilai defaultnya adalah 53. Bila digunakan port lain ada kemungkinan aplikasi standard tidak bisa mengakses **named**.

-b *bootfile* Menentukan bootscript yang digunakan saat menjalankan **named**. File defaultnya adalah **/etc/named.boot**.

Perintah berikut akan menjalankan named dengan menggunakan nilai default:

```
#/etc/named
```

Bila digunakan bootscript selain named.boot (atau jika direktori yang digunakan berbeda), dapat digunakan:

```
#/etc/named -b bootfile
```

4.6. Menkonfigurasi Resolver

Resolver adalah program yang mengirim pertanyaan (query) terhadap Server DNS untuk mendapatkan informasi tentang suatu host. Dalam sistem operasi UNIX resolver diimplementasikan dalam suatu library (bukan dalam suatu program khusus untuk client). Untuk menggunakan resolver user cukup mengkonfigurasi file /etc/resolv.conf. File ini berisi informasi tentang domain dan Server DNS untuk domain tersebut. Berikut ini adalah salah satu contoh file /etc/resolv.conf.

```
;
; Contoh file /etc/resolv.conf
;
domain alexandria.cnrg.net
server 132.92.120
domain luxor.cnrg.net
server 132.92.120
```

Keterangan:

Untuk domain *alexandria.cnrg.net*, Server DNS-nya adalah 132.92.22.120

Untuk domain *luxor.cnrg.net*, Server DNSnya adalah 132.92.31.132

Untuk sebuah domain bisa digunakan lebih dari satu Server DNS

4.7. Memelihara Server DNS

Setelah Server DNS diaktifkan, ia perlu dipelihara dan dirawat. Ada beberapa hal yang harus dilakukan untuk memelihara Server DNS, antara lain:

Setiap melakukan perubahan pada zona file **jangan lupa** menambah serial number pada SOA Record. Karena apabila serial number tidak diubah maka secondary server tidak akan melakukan zona transfer ke primary server.

Sebaiknya format serial number yang digunakan adalah: YYMMddhhmm (YY=dua angka terakhir dari tahun, MM=bulan, dd=tanggal, hh=jam, mm=menit). Artinya bahwa setiap ada perubahan pada zona file maka serial number diganti dengan angka-angka tahun, bulan, tanggal, jam dan menit tanggal perubahan. Dengan cara ini maka setiap ada perubahan maka serial number pasti bertambah besar.

Untuk kehandalan system sebaiknya selain primary server juga digunakan beberapa secondary server. Sehingga apabila primary server mengalami kegagalan maka pemakai dapat menggunakan secondary server sebagai name server. Usahakan melakukan koordinasi dengan administrator yang menangani primary server dari berbagai domain yang saling terhubung.

Untuk mendapatkan file db.cache release terakhir, kirim mail ke **vice@nic.ddn.mil** dengan subject **netinfo root-servers.txt**. Update file db.cache dengan informasi root server yang terakhir.

4.8. Tip Konfigurasi dan Utilitas DNS

Menjalankan name server bukanlah hal yang mudah. Banyak hal yang harus dilakukan, dan sangat mudah terjadi kesalahan. Karenanya, diperlukan kehati-hatian yang cukup tinggi dalam menginstalasi name server agar ia berjalan dengan baik. Pada subbab ini akan diberikan beberapa tips dalam mengatasi kesalahan konfigurasi Server DNS serta tools yang dilakukan untuk mendebug Server DNS. Sebagian informasi ini diambil dari RFC 1912 yang membahas hal di atas.

4.8.1. Tip dalam mengkonfigurasi DNS

Penamaan host

Dalam penamaannya, sebuah domain terdiri atas beberapa kata /label yang dipisahkan oleh tanda titik. Ada beberapa aturan yang harus diikuti dalam memilih nama host atau domain, agar dalam pengoperasian DNS Anda tidak bermasalah.

Karakter yang digunakan hanya boleh terdiri atas huruf dan angka yang tercantum kode ASCII, serta tanda "-"(*dash*). Label ini tidak boleh semuanya terdiri atas angka, dan hanya boleh berakhir dengan angka atau huruf.

Penggunaan tanda *underscore* ("_") sebaiknya dihindari. Hal ini perlu dilakukan mengingat terdapat implementasi TCP/IP yang menolak nama host yang mengandung tanda *underscore* ini.

Karena aturan nama host ini juga dipakai untuk pengiriman email, maka aturan yang valid untuk penamaan host untuk pengiriman email (RFC 822) juga berlaku untuk penamaan host

Jangan memilih nama host yang bisa menimbulkan interpretasi yang salah bagi library penerjemah hostname (misalnya fungsi **inet_ntoa**() di sistem operasi unix). Misalkan Anda menggunakan nama host **0xe**. Nama ini valid menurut aturan penamaan host. Namun Jika Anda mengetikkan **telnet 0xe**, maka perintah telnet di atas akan di interpretasikan oleh fungsi inet_ntoa() di atas sebagai **telnet 0.0.0.14**, karena 0xe merupakan penulisan dari E (=14) heksadesimal.

Glue record dan bahayanya

Glue record ialah Address record yang diasosiasikan dengan NS record tertentu. Contoh dari *glue record* adalah sebagai berikut.

```
alexandria.cnrg.net.        IN  NS  snefera.alexandria.cnrg.net.
                            IN  NS  cheops.alexandria.cnrg.net.
snefera.alexandria.cnrg.net.  IN  A     132.92.129.3
cheops.alexandria.cnrg.net.   IN  A     132.92.36.123
```

Pada contoh di atas, IN A record disebut sebagai *glue record*. Fungsinya ialah menunjukkan IP address name server yang digunakan oleh domain yang didelegasikan.

Ada beberapa hal yang perlu diperhatikan dalam menuliskan *glue record* ini. Pertama, *glue records* hanya diperlukan oleh zona file *forward. Glue record* tidak diperlukan dalam zona file untuk reverse domain.

Kedua, jika name server yang bersangkutan adalah multihomed (memiliki lebih dari satu IP address), maka kedua buah IP address tersebut harus dituliskan dalam *glue record*. Hal ini dilakukan untuk menghindari inkonsistensi cache.

Ketiga, jangan biasakan untuk selalu membuat *glue record* tiap kali kita membuat NS record. Satu *glue record* sudah cukup. *Glue record* yang terduplikasi akan menjadi masalah pada saat Anda melakukan penggantian IP address name server Anda. IP yang lama akan tetap muncul karena Anda lupa bahwa dibagian lain zona file atau di zona file yang lain masih terdapat record dengan IP address yang lama. Hal ini akan semakin menjadi masalah jika data yang salah tersebut dipropagasikan bolak balik antara primary dan secondary server. Untuk menanggulangi hal ini, kedua server harus dimatikan, dihapus file backup DNSnya dan semua Server DNS tersebut di restart lagi.

Inkonsistensi record (antara PTR dengan A)

Setiap host yang terhubung ke Internet harus memiliki nama. Sebagian besar service di Internet menggunakan DNS untuk memeriksa layak tidaknya ia memberikan service kapada Anda. Cukup banyak service di Internet yang menolak memberikan service jika host Anda tidak terdaftar secara benar pada Server DNS. Contohnya ialah FTP server & IRC server yang menolak memberikan service jika host Anda memiliki *forward* dan *reverse address* yang berbeda.

Karenanya, pastikan PTR record dan A record di Server DNS Anda sesuai. Untuk setiap IP address, harus terdapat PTR record yang cocok dalam domain

in-addr.arpa. jika host tersebut multihomed (memiliki lebih dari satu IP address), seluruh IP addresnya harus memiliki PTR record. Selain itu PTR record harus menunjuk kembali ke A record yang sah, dan tidak menunjuk ke CNAME record.

Kesalahan penggunaan CNAME untuk MX dan NS

Record CNAME sebaiknya hanya digunakan mengeneralisasi nama server (www untuk web server, ftp untuk ftp server dan sebagainya). Record CNAME sama sekali tidak diperbolehkan untuk digunakan bersama-sama resource record yang lain.

Sebagai contoh, jika sudah didefinisikan bahwa *mail. alexandria.cnrg.net* merupakan alias dari *nefertiti. alexandria.cnrg.net*, tidak boleh ada MX record, NS record atau record apapun untuk nama alias ini. Jangan sekali-kali melakukan praktek seperti yang ditunjukkan di bawah ini:

```
alexandria.cnrg.net.    IN  MX       mail
mail                    IN  CNAME    nefertiti
nefertiti               IN  A        132.92.36.23
```

Hal di atas hanya akan memperberat kerja server dan resolver karena untuk me-resolve host mail.alexandria. cnrg.net akan diperlukan dua kali pemrosesan.

Yang lebih berbahaya ialah menggunakan record NS bersama-sama dengan CNAME, sebagaimana ditunjukkan oleh contoh di bawah ini

```
Alexandria.cnrg.net.  IN  NS  snefera.alexandria.cnrg.net
                      IN  NS  cheops.alexandria.cnrg.net.
                      IN  CNAME      snefera
Snefera               IN  A   132.92. 129.3
```

Contoh di atas menunjukkan cara yang salah untuk membuat nama domain menjadi nama host (misalnya *telnet alexandria.cnrg.net* = *telnet snefera .alexandria.cnrg.net*). Karena CNAME tidak diperbolehkan digunakan bersama-sama record yang lain, implementasi Server DNS (dalam hal ini *BIND*) akan menolak membaca record lain setelah melihat NS record menunjuk ke CNAME. Dengan demikian, seluruh resource record yang lain, termasuk IN NS dan IN A akan ditolak, sehingga seluruh nama host dalam zona file akan menjadi sia-sia.

Jika Anda ingin agar nama domain menjadi nama host, lakukanlah dengan cara berikut ini.

```
Alexandria.cnrg.net.  IN  NS  snefera.alexandria.cnrg.net
                      IN  NS  cheops.alexandria.cnrg.net.
                      IN  A   132.92.129.3
Snefera               IN  A   132.92.129.3
```

Perlu diperhatikan, jangan lupa untuk meng-hapus CNAME yang diasosiasikan dengan hostname tertentu bila Anda menghapus host tersebut dari zona file.

Lame delegation

Setiap domain diharuskan untuk paling tidak memiliki primary server dan satu secondary server. Secondary server ini sebaiknya berada diluar network Anda. Jika secondary server ini tidak langsung berada di bawah tanggung jawab Anda, lakukan pemeriksaan isinya secara periodik untuk memastikan mereka mendapatkan data yang terbaru hasil zona transfer. Jika secondary server ini kita beri pertanyaan, ia harus memberikan jawaban yang *authoritative*. Jika jawaban yang diberikan tidak authoritative, hal ini termasuk kasus *Lame Delegation*.

Lame Delegation ialah kasus dimana kita sudah mendelegasikan suatu server untuk menjadi primary atau secondary server, namun server yang bersangkutan tidak melakukan tugasnya dalam menyediakan name service untuk zona yang bersangkutan.

```
Alexandria.cnrg.net.   IN  NS  snefera.alexandria.cnrg.net.
IN      NS cheops.alexandria.cnrg.net.
```

Seperti contoh di atas, jika pada named.boot kita sudah menuliskan hal seperti di atas, namun secondary server *cheops* belum siap (belum melakukan zona transfer, atau belum di set sebagai secondary), maka akan terjadi *lame delegation*. Banyak hal akan terjadi sebagai akibat *dari lame delegation*. Dari mulai trafik tambahan ke primary server, nama host yang tidak dapat di resolve, hingga mentalnya e-mail ke host tujuan.

4.8.2. Utilitas DNS

Nslookup

Nslookup ialah program untuk meng-query server DNS untuk mendapatkan informasi yang diinginkan. Program ini sangat berguna untuk meme-riksa benar tidaknya data yang terdapat di Server DNS. Terdapat dua mode dalam penggunaan nslookup ini, yaitu mode interaktif dan mode command line (non-interaktif.)

Mode Noninteraktif

Mode ini digunakan jika dalam menjalankan program nslookup, Anda memasukkan nama host atau IP address sebagai argumen. Argumen kedua bisa berupa nama host atau alamat Server DNS.

Contoh:

```
> nslookup mx1.itb.ac.id
Server:  ns2.ITB.ac.id
Address:  167.205.22.123
Name:    mx1.itb.ac.id
Addresses: 202.249.47.36, 167.205.23.6
>
```

Pada contoh di atas, query diajukan ke name server ns1.itb.ac.id. jawaban atas query menunjukkan mx1.itb.ac.id memiliki dua buah IP address, yaitu 202.249.47.36 dan 167.205.23.6

Mode Interaktif

Untuk memasuki mode interaktif, cukup diketikkan `nslookup` tanpa argumen berupa nama host atau IP address. Untuk keluar dari mode interaktif ini, Anda cukup mengetikkan control-D atau mengetikkan "`exit`". Untuk membatalkan perintah perintah yang Anda ketikkan, Anda dapat mengetikkan control-C.

Jika Anda ingin mengetikkan nama host yang kebetulan sama dengan perintah built-in dari nslookup, Anda harus menambahkan karakter "\" sebelum mengetikkan nama host tersebut. Jika Anda mengetikkan perintah yang tidak dimengerti oleh nslookup, perintah ini akan diterjemahkan sebagai nama host.

Contoh:

```
> nslookup
Default Server:  ns2.ITB.ac.id
Address:  167.205.22.123
> ?
$Id: nslookup.help,v 8.3 1996/08/05 08:31:39 vixie Exp $
Commands: (identifiers are shown in uppercase, [] means optional)
NAME - print info about the host/domain NAME using default server
```

```
NAME1 NAME2    - as above, but use NAME2 as server
help or ? - print info on common commands; see nslookup(1) for details
set OPTION     - set an option
   all         - print options, current server and host
   [no]debug   - print debugging information
   [no]d2      - print exhaustive debugging information
   [no]defname - append domain name to each query
   [no]recurse - ask for recursive answer to query
   [no]vc      - always use a virtual circuit
   domain=NAME - set default domain name to NAME
   srchlist=N1[/N2/.../N6] - set domain to N1 and search list to N1,N2, etc.
   root=NAME   - set root server to NAME
   retry=X     - set number of retries to X
   timeout=X   - set initial time-out interval to X seconds
   querytype=X - set query type, e.g.,
A,ANY,CNAME,HINFO,MX,PX,NS,PTR,SOA,TXT,WKS
   port=X      - set port number to send query on
   type=X      - synonym for querytype
   class=X - set query class to one of IN (Internet), CHAOS,HESIOD or ANY
server NAME    - set default server to NAME, using current default server
lserver NAME   - set default server to NAME, using initial server
finger [USER]  - finger the optional USER at the current default host
root           - set current default server to the root
ls [opt] DOMAIN[>FILE] - list addresses in DOMAIN (optional: output to FILE)
    -a         - list canonical names and aliases
    -h         - list HINFO (CPU type and operating system)
    -s         - list well-known services
    -d         - list all records
    -t TYPE  - list records of the given type (e.g., A,CNAME,MX, etc.)
view FILE      - sort an 'ls' output file and view it with more
exit           - exit the program, ^D also exits
```

Seluruh perintah yang dimengerti oleh nslookup dapat ditampilkan dengan mengetikkan "?".

Mencari Authoritative server dari suatu domain

```
> nslookup
Default Server:  ns2.ITB.ac.id
Address:  167.205.22.123
> www.lycos.com
Server:  ns2.ITB.ac.id
```

```
Address: 167.205.22.123
Name:    www.lycos.com
Addresses: 207.77.90.13, 207.77.90.14, 207.77.90.15, 208.213.76.35
207.77.90.12
```

Saat kita mengetikkan perintah di atas, secara default kita memaksa name server kita (ns2.itb.ac.id) melakukan resolution secara rekursif. Jawaban yang dihasilkannya pun authoritative.

```
> set norecurse
> www.lycos.com
Server: ns2.ITB.ac.id
Address: 167.205.22.123
Non-authoritative answer:
Name:    www.lycos.com
Addresses: 207.77.90.14, 207.77.90.15, 208.213.76.35, 207.77.90.12
    207.77.90.13
>
```

Hasil yang berbeda akan kita dapatkan kita tidak memperbolehkan server melakukan resolution dengan mengikuti pointer-pointer yang diberikan oleh name server authoritative. Ns2.itb.ac.id akan memberikan jawaban non authoritative yang berasal dari cache-nya.

```
> set recurse
> set type=NS
> www.lycos.com
Server: ns2.ITB.ac.id
Address: 167.205.22.123
lycos.com
        origin = ns.lycos.com
        mail addr = networking.lycos.com
        serial = 980651255
        refresh = 900 (15 mins)
        retry  = 1800 (30 mins)
        expire  = 604800 (7 days)
        minimum ttl = 300 (5 mins)
```

dengan memilih jenis query NS, kita dapat mengetahui siapa sebenarnya name server yang authoritative untuk domain lycos.com

Mencari IP address dari hostname

```
> server ns.lycos.com
Default Server: ns.lycos.com
Address: 207.77.91.13
> set type=A
> www.lycos.com
Server: ns.lycos.com
Address: 207.77.91.13
Name:   www.lycos.com
Addresses: 207.77.90.15, 208.213.76.35, 207.77.90.12, 207.77.90.13,
207.77.90.14
```

Dengan mengetikkan perintah server ns.lycos.com, kita membuat nslookup langsung berhubungan dengan Server DNS authoritative dari domain lycos.com.

Perintah berikutnya, set type=A, mengeset jenis query yang akan kita ajukan, yaitu A record. Dengan mengetikkan **www.lycos.com**, berarti kita meminta informasi ke name server tentang IP address **www.lycos.com.** terlihat bahwa **www.lycos.com** memiliki 5 IP address. Lima IP address ini belum tentu berarti satu host memiliki lima interface. Untuk kepentingan lycos (search engine dan pembagian beban), hal ini berarti lima host dengan IP berbeda memiliki hostname yang sama.

Mencari hostname dari IP address

```
> set type=PTR
> 207.77.90.15
Server: ns.lycos.com
Address: 207.77.91.13
15.90.77.207.in-addr.arpa      name = www.lycos.com
90.77.207.in-addr.arpa  nameserver = ns1.lycos.com
```

```
90.77.207.in-addr.arpa  nameserver = ns2.lycos.com
ns1.lycos.com   internet address = 207.77.90.10
ns2.lycos.com   internet address = 207.77.90.11
```

Perintah set type = PTR akan membuat jenis query menjadi PTR. Jika Anda menuliskan IP address, akan dihasilkan jawaban berupa hostname. Jika tidak, akan dihasilkan petunjuk ke informasi lain.

Mencari SOA dari suatu domain

```
> set type=SOA
> www.lycos.com
Server:  ns.lycos.com
Address:  207.77.91.13
lycos.com
        origin = ns.lycos.com
        mail addr = networking.lycos.com
        serial = 980651255
        refresh = 900 (15 mins)
        retry   = 1800 (30 mins)
        expire  = 604800 (7 days)
        minimum ttl = 300 (5 mins)
>
```

Perintah set type=SOA akan menghasilkan terhadap SOA record.

Mencari MX dari suatu host atau domain

```
> set type=MX
> www.lycos.com
Server:  ns.lycos.com
Address:  207.77.91.13
www.lycos.com   preference = 10, mail exchanger =
mailhost.lycos.com
lycos.com       nameserver = ns2.lycos.com
lycos.com       nameserver = ns1.lycos.com
lycos.com       nameserver = ns2.mci.net
mailhost.lycos.com      internet address = 208.200.146.250
```

```
ns2.lycos.com   internet address = 207.77.90.11
ns1.lycos.com   internet address = 207.77.90.10
ns2.mci.net   internet address = 204.70.57.242
>
```

Untuk mendapatkan informasi tentang mail exchanger dari suatu domain, diketikkan set type=MX. Pada contoh di atas, untuk **www.lycos.com** ternyata terdapat satu alamat yang berfungsi sebagai mail exchanger, yaitu mailhost.lycos.com.

Mencari informasi lengkap tentang suatu host

```
> set type=ANY
> www.lycos.com
Server:  ns.lycos.com
Address:  207.77.91.13
www.lycos.com   internet address = 207.77.90.15
www.lycos.com   internet address = 208.213.76.35
www.lycos.com   internet address = 207.77.90.12
www.lycos.com   internet address = 207.77.90.13
www.lycos.com   internet address = 207.77.90.14
www.lycos.com   preference = 10, mail exchanger = mailhost.lycos.com
lycos.com       nameserver = ns2.lycos.com
lycos.com       nameserver = ns1.lycos.com
lycos.com       nameserver = ns2.mci.net
mailhost.lycos.com      internet address = 208.200.146.250
ns2.lycos.com   internet address = 207.77.90.11
ns1.lycos.com   internet address = 207.77.90.10
ns2.mci.net     internet address = 204.70.57.242
>
```

Dengan membuat jenis query ANY, berarti kita meminta informasi seluruh record yang cocok dengan nama host yang kita inginkan.

4.9. Ringkasan

DNS digunakan untuk mempermudah penggunaan Internet, dengan memetakan IP address ke nama host. Agar data nama host dapat di distribusikan di banyak server, format data yang digunakan harus mencerminkan terdistribusinya data tersebut. Untuk itu, digunakan format tree dengan masing-masing nodenya disebut domain. Penulisan nama host secara lengkap disebut sebagai Full Qualified Domain Name (FQDN)

DNS terdiri atas beberapa komponen, yaitu server, resolver, dan cache. Server berfungsi menyediakan jawaban atas pertanyaan domain, resolver berfungsi mencari jawaban berdasarkan pertanyaan domain dari aplikasi, dan cache berfungsi menyimpan sementara data hasil query untuk menghemat waktu proses *name resolution*

Software server DNS yang paling banyak digunakan ialah BIND. Dalam menjalankan BIND diperlukan konfigurasi atas boot script dan zona file. Boot script ialah sekumpulan file yang diperlukan untuk menginisialisasi operasi server DNS. Sedangkan zona file berisi data nama domain dari jaringan tertentu.

Untuk keperluan redundansi, server DNS untuk satu domain dapat lebih dari satu. Server utama dinamai *primary server*, sedangkan yang lain disebut *secondary server*. Zona file cukup diletakkan di primary server. *Secondary server* akan mengambil data ini dari *primary server* melalui proses yang disebut *zona transfer*.

Selain memetakan nama host ke IP address, DNS juga mampu melakukan hal sebaliknya. Proses pemetaan ini

disebut reverse mapping, sedangkan datanya disebut sebagai reverse domain. Seluruh informasi reverse domain ini diletakkan di bawah domain **in-addr.arpa**

Bab 5
Routing di Jaringan TCP/IP

Tujuan dibuatnya jaringan komputer adalah untuk menyampaikan data dari satu komputer ke komputer lain. Jika jaringan tidak dapat menyampaikan data ke tujuan yang seharusnya, maka jaringan tersebut tidak berguna. Pada bab sebelum ini kita sudah mengetahui bahwa protokol TCP/IP memiliki kemampuan untuk internetworking dan routing. Dalam jaringan TCP/IP setiap host memiliki IP address dan untuk berhubungan dengan host tersebut kita harus memasukkan IP address host pada bagian tujuan dari datagram IP yang dikirim. Pertanyaan berikutnya yang harus dapat dijawab adalah bagaimana host mengetahui cara mencapai host yang dituju datagram tersebut?

Bab ini membahas mengenai proses yang dialami datagram untuk mencapai tujuan di jaringan TCP/IP, yaitu routing. Pertama-tama akan dibahas konsep dasar routing, kemudian jenis-jenis routing, tabel routing, protokol routing serta bagaimana protokol routing tersebut bekerja.

5.1. Konsep Dasar Routing

Konsep routing adalah hal yang utama pada lapisan internet di jaringan TCP/IP. Hal ini karena pada lapisan internet terjadi pengalamatan (*addressing*). Kita coba perhatikan kembali aliran data pada arsitektur TCP/IP. Data dari lapisan aplikasi disampaikan ke lapisan transport dengan diberi header TCP atau UDP tergantung jenis aplikasinya. Setelah itu segmen TCP atau UDP disampaikan ke lapisan IP dan diberi header, termasuk alamat asal dan tujuan datagram. Pada saat ini host harus melakukan routing dengan melihat tabel routing. Setelah melihat tabel routing, datagram diteruskan ke lapisan network interface dan diberi header dengan alamat tujuan yang sesuai.

Untuk lebih jelasnya, kita perhatikan jaringan TCP/IP yang menggunakan teknologi Ethernet. Setiap frame Ethernet (Ethernet II) mengandung alamat tujuan dan alamat asal, tipe protokol, dan data. Alamat tujuan dan asal adalah sebuah bilangan 48 bit. Setiap kartu Ethernet memiliki sebuah alamat Ethernet yang unik. Agar datagram dapat diterima oleh sebuah host tujuan, datagram tersebut harus dimasukkan dalam frame dengan alamat Ethernet tujuan yang sama dengan alamat kartu Ethernet host tujuan. Proses ini juga bagian dari routing, yaitu pada saat mengirimkan datagram IP bagaimana menentukan alamat Ethernet host tujuan datagram tersebut?

5.1.1. ARP

Pertanyaan ini dijawab dengan adanya protokol ARP (*Address Resolution Protocol*). ARP bertugas untuk

menerjemahkan IP address ke alamat Ethernet. Proses ini dilakukan hanya untuk datagram yang dikirim host karena pada saat inilah host menambahkan header Ethernet pada datagram. Penerjemahan dari IP address ke alamat Ethernet dilakukan dengan melihat sebuah tabel yang disebut sebagai *cache ARP*, lihat Tabel 5.1. Entri cache ARP berisi IP address host beserta alamat Ethernet untuk host tersebut. Tabel ini diperlukan karena tidak ada hubungan sama sekali antara IP address dengan alamat Ethernet. IP address suatu host bergantung pada IP address jaringan tempat host tersebut berada, sementara alamat Ethernet sebuah kartu bergantung pada alamat yang diberikan oleh pembuatnya.

Tabel 5.1 Cache ARP

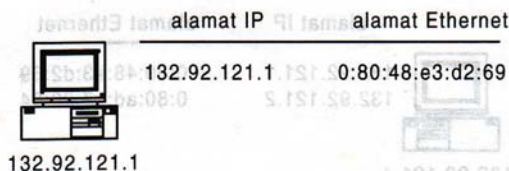
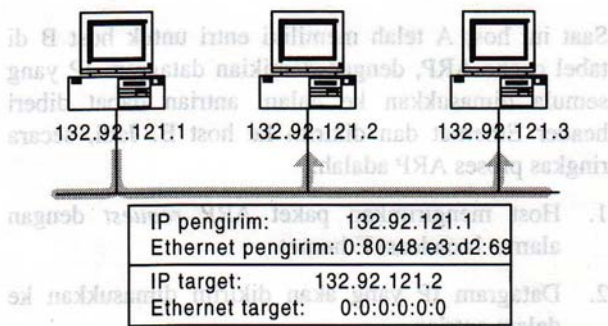
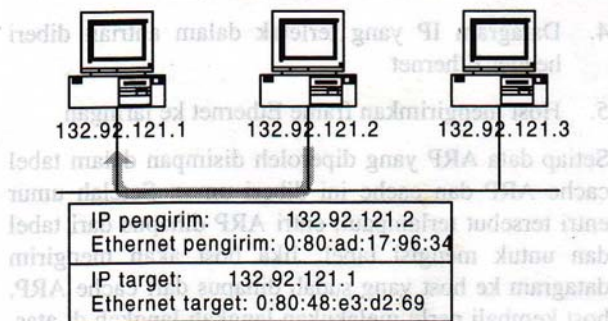
IP address	Alamat Ethernet
132.92.121.1	0:80:48:e3:d2:69
132.92.121.2	0:80:ad:17:96:34
132.92.121.3	0:20:4c:30:29:29

Mekanisme penerjemahan oleh ARP dapat dijelaskan sebagai berikut. Misal suatu host A dengan IP address 132.92.121.1 baru dinyalakan, lihat Gambar 5.1. Pada saat awal, host ini hanya mengetahui informasi mengenai interface-nya sendiri, yaitu IP address, alamat network, alamat broadcast, dan alamat Ethernet. Dari informasi awal ini, host A tidak mengetahui alamat Ethernet host lain yang terletak satu network dengannya (cache ARP hanya berisi satu entri, yaitu host A). Jika host memiliki rute default, maka entri yang pertama kali dicari oleh ARP adalah router default tesebut.

Misalkan terdapat datagram IP dari host A yang ditujukan kepada host B yang memiliki IP 167.205.121.2 (host B ini terletak satu subnet dengan host A). Saat ini

yang diketahui oleh host A adalah IP address host B tetapi alamat Ethernet host B tidak diketahui. Agar dapat mengirimkan datagram ke host B, host A perlu mengisi cache ARP dengan entri host B. Karena cache ARP tidak dapat digunakan untuk menerjemahkan IP address host B menjadi alamat Ethernet, maka host A harus melakukan dua hal:

- mengirimkan paket ARP *request* pada seluruh host di network menggunakan alamat broadcast Ethernet (FF:FF:FF:FF:FF:FF) untuk meminta jawaban ARP dari host B, lihat Gambar 5.2.
- menempatkan datagram IP yang hendak dikirim dalam antrian.

Paket ARP *request* yang dikirim host A kira-kira berbunyi "Jika IP address-mu adalah 167.205.121.2, mohon beritahu alamat Ethernet-mu". Karena paket ARP *request* dikirim ke alamat broadcast Ethernet, setiap interface Ethernet komputer dalam jaringan dapat mendengarnya. Setiap host dalam jaringan tersebut kemudian memeriksa apakah IP addressnya sama dengan IP address yang diminta oleh host A. Host B yang mengetahui bahwa yang diminta oleh host A adalah IP address yang dimilikinya langsung memberikan jawaban dengan mengirimkan paket ARP *response* langsung ke alamat Ethernet pengirim (host A), seperti terlihat pada Gambar 5.3. Paket ARP *response* kira-kira berbunyi "IP address 167.205.121.2 adalah milik saya, sekarang saya berikan alamat Ethernet saya." Paket ARP *response* dari host B tersebut diterima oleh host A dan host A kemudian menambahkan entri IP address host B beserta alamat Ethernet-nya ke dalam cache ARP, lihat Gambar 5.4.

Gambar 5.1 Cache ARP awal

Gambar 5.2 Paket ARP request

Gambar 5.3 Paket ARP response

	alamat IP	alamat Ethernet
132.92.121.1	132.92.121.1	0:80:48:e3:d2:69
	132.92.121.2	0:80:ad:17:96:34

Gambar 5.4 Cache ARP setelah penambahan entri host B

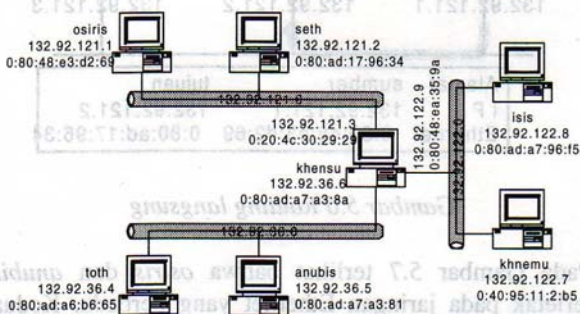
Saat ini host A telah memiliki entri untuk host B di tabel cache ARP, dengan demikian datagram IP yang semula dimasukkan ke dalam antrian dapat diberi header Ethernet dan dikirim ke host B. Jadi, secara ringkas proses ARP adalah:

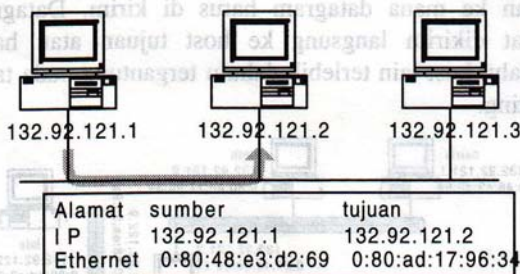
1. Host mengirimkan paket ARP *request* dengan alamat broadcast Ethernet
2. Datagram IP yang akan dikirim dimasukkan ke dalam antrian
3. Paket ARP *response* diterima host dan host mengisi tabel ARP dengan entri baru
4. Datagram IP yang terletak dalam antrian diberi header Ethernet
5. Host mengirimkan frame Ethernet ke jaringan

Setiap data ARP yang diperoleh disimpan dalam tabel cache ARP dan cache ini diberi umur. Setelah umur entri tersebut terlampaui, entri ARP dihapus dari tabel dan untuk mengisi tabel. Jika host akan mengirim datagram ke host yang sudah dihapus dari cache ARP, host kembali perlu melakukan langkah-langkah di atas. Dengan cara ini dimungkinkan terjadinya perubahan isi

cache ARP yang dapat menunjukkan dinamika jaringan. Jika sebuah host di jaringan dimatikan, maka selang beberapa saat kemudian entri ARP untuk host tersebut dihapus karena kadaluwarsa. Jika kartu ethernet host tersebut diganti, maka beberapa saat kemudian entri ARP host berubah dengan informasi alamat Ethernet yang baru.

5.2. Routing Langsung dan Tidak Langsung

Seperti telah disebutkan di atas, proses pengiriman datagram IP selalu menggunakan tabel routing. Tabel routing berisi informasi yang diperlukan untuk menentukan ke mana datagram harus di kirim. Datagram dapat dikirim langsung ke host tujuan atau harus melalui host lain terlebih dahulu tergantung pada tabel routing.

Gambar 5.5 Jaringan TCP/IP

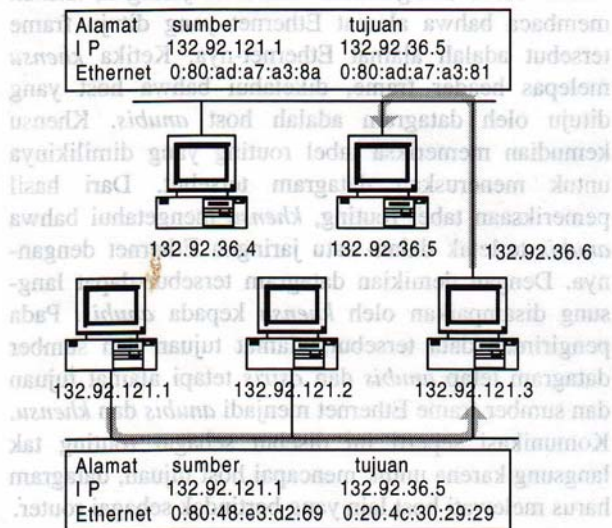
Gambar 5.5 memperlihatkan jaringan TCP/IP yang menggunakan teknologi Ethernet. Pada jaringan tersebut jika host *osiris* mengirimkan data ke host *seth*, alamat tujuan datagram adalah *seth* dan alamat sumber datagram adalah *osiris*. Frame yang dikirimkan oleh host *osiris* juga memiliki alamat tujuan frame *seth* dan alamat sumbernya adalah *osiris*. Pada saat *osiris* mengirimkan frame, *seth* membaca bahwa frame tersebut ditujukan kepada alamat Ethernetnya. Setelah melepas header frame, *seth* kemudian mengetahui bahwa IP address tujuan datagram tersebut juga adalah IP address-nya. Dengan demikian *seth* meneruskan datagram ke lapisan transport untuk diproses lebih lanjut. Komunikasi model seperti ini disebut sebagai routing langsung.

Gambar 5.6 Routing langsung

Pada Gambar 5.7 terlihat bahwa *osiris* dan *anubis* terletak pada jaringan Ethernet yang berbeda. Kedua jaringan tersebut dihubungkan oleh *khensu*. *Khensu* memiliki lebih dari satu interface dan dapat melewatkan datagram dari satu interface ke interface yang lain (bertindak sebagai router). Ketika mengirimkan data ke *anubis*, *osiris* memeriksa tabel routing dan mengetahui

bahwa data tersebut harus melewati *khensu* terlebih dahulu. Dengan kondisi seperti ini datagram yang dikirim *osiris* ke *anubis* memiliki alamat tujuan *anubis* dan alamat sumber *osiris* tetapi frame Ethernet yang dikirimnya diberi alamat tujuan *khensu* dan alamat sumber *osiris*.

Ketika *osiris* mengirimkan frame ke jaringan, *khensu* membaca bahwa alamat Ethernet yang dituju frame tersebut adalah alamat Ethernet-nya. Ketika *khensu* melepas header frame, diketahui bahwa host yang dituju oleh datagram adalah host *anubis*. Khensu kemudian memeriksa tabel routing yang dimilikinya untuk meneruskan datagram tersebut. Dari hasil pemeriksaan tabel routing, *khensu* mengetahui bahwa *anubis* terletak dalam satu jaringan Ethernet dengan-nya. Dengan demikian datagram tersebut dapat lang-sung disampaikan oleh *khensu* kepada *anubis*. Pada pengiriman data tersebut, alamat tujuan dan sumber datagram tetap *anubis* dan *osiris* tetapi alamat tujuan dan sumber frame Ethernet menjadi *anubis* dan *khensu*. Komunikasi seperti ini disebut sebagai routing tak langsung karena untuk mencapai host tujuan, datagram harus melewati host lain yang bertindak sebagai router.

Pada dua kasus di atas terlihat proses yang terjadi pada lapisan internet ketika mengirimkan dan menerima da-tagram. Pada saat mengirimkan datagram, host harus memeriksa apakah alamat tujuan datagram terletak pada jaringan yang sama atau tidak. Jika alamat tujuan datagram terletak pada jaringan yang sama, datagram dapat langsung disampaikan. Jika ternyata alamat tu-juan datagram tidak terletak pada jaringan yang sama, datagram harus disampaikan melalui host lain yang bertindak sebagai router. Pada saat menerima datagram

host harus memeriksa apakah ia merupakan tujuan dari datagram tersebut. Jika memang demikian maka data diteruskan ke lapisan transport. Jika ia bukan tujuan dari datagram tersebut, maka datagram tersebut dibuang. Jika host yang menerima datagram bertindak sebagai router, maka ia meneruskan datagram ke interface yang menuju alamat tujuan datagram.

Gambar 5.7 Routing tak langsung

5.3. Tabel Routing

Di atas telah dijelaskan bagaimana cara host menyampaikan datagram. Penyampaian datagram tersebut selalu menggunakan tabel routing dalam menjalankan prosesnya. Tabel routing terdiri atas entri-entri rute

dan setiap entri rute setidaknya terdiri atas IP address, tanda untuk menunjukkan routing langsung atau tak langsung, alamat router, dan nomor interface, lihat Tabel 5.2. IP address pada entri tabel routing dapat berupa IP address jaringan atau IP address host. Perbedaan jenis IP address ini dapat diketahui dengan melihat subnet mask bagi tabel routing tersebut. Tanda routing langsung/tak langsung menentukan alamat router. Pada routing tak langsung, alamat router adalah IP address host lain yang dapat menyampaikan datagram ke tujuannya. Nomor interface pada entri tabel routing menunjukkan jalur keluar datagram dari host.

Tabel 5.2 Contoh tabel routing

IP address	Langsung/tidak	Router	No. interface
132.92.36	Langsung	<kosong>	1
132.92.122	tak langsung	132.92.36.6	1

Sekarang kita akan perhatikan jaringan TCP/IP pada Gambar 5.5 untuk melihat tabel routing bekerja. Pada gambar terlihat tiga jaringan ethernet yang dihubungkan oleh sebuah router dengan tiga interface. Host-host pada jaringan tersebut dibuatkan tabel routing sehingga dapat menjangkau seluruh host yang lain.

Tabel 5.3 Tabel routing host isis

IP address	Langsung/tidak	Router	No. Interface
132.92.122	langsung	<kosong>	1
132.92.121	tak langsung	132.92.122.9	1
132.92.36	tak langsung	132.92.122.9	1

Misal *isis* hendak mengirim data ke *toth*. Pada lapisan internet, *isis* memeriksa alamat network dari *toth*. Kemudian alamat network tersebut dibandingkan dengan yang terdapat di tabel routing dan diperoleh

bahwa alamat network sama dengan baris ketiga tabel routing, lihat Tabel 5.3. Agar dapat sampai ke *toth* data harus dikirim melalui *khensu* terlebih dahulu. Host *isis* kemudian menerjemahkan IP address *khensu* menjadi alamat Ethernetnya. Setelah memperoleh hasil proses ARP, datagram kemudian diberi header frame dan frame dikirim melalui interface nomor 1.

Tabel 5.4 Tabel routing khensu

IP address	Langsung/tidak	Router	No. Interface
132.92.122	langsung	<kosong>	1
132.92.121	langsung	<kosong>	2
132.92.36	langsung	<kosong>	3

Host *khensu* menerima frame pada interface nomor 1. Frame ini kemudian dilepas dan host *khensu* memeriksa kembali alamat tujuan datagram. Karena alamat tujuan datagram tersebut bukan *khensu* dan *khensu* bertindak sebagai router, maka datagram tersebut akan diteruskan ke host tujuan. Proses pembandingan alamat tujuan datagram dengan alamat di tabel routing seperti di *osiris* berulang lagi. Pada proses ini, *khensu* menemukan bahwa host *toth* terletak satu jaringan dengannya (baris ketiga tabel routing, lihat Tabel 5.4). Dengan demi-kian datagram dapat langsung dibungkus dengan frame yang diberi alamat tujuan host *toth* dan dikirimkan melalui interface nomor 3. Frame tersebut akhirnya diterima oleh host *toth* dan karena melihat bahwa tujuan datagram adalah host *toth*, maka datagram tersebut diteruskan ke lapisan transport TCP/IP.

Proses pencarian pada tabel routing ini biasanya mengikuti langkah-langkah di bawah ini:

1. Alamat tujuan datagram di-masking dengan subnet mask host pengirim dan dibandingkan dengan alamat network host pengirim. Jika sama, maka ini adalah routing langsung dan frame langsung dikirimkan ke interface jaringan.

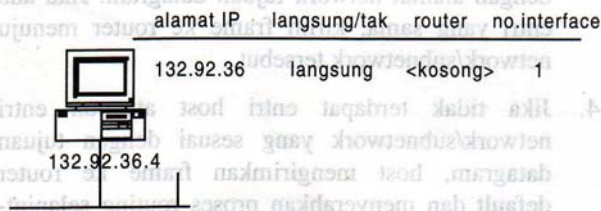
2. Jika tujuan datagram tidak terletak dalam satu jaringan, periksa apakah terdapat entri routing yang berupa host dan bandingkan dengan IP address tujuan datagram. Jika ada entri yang sama, kirim frame ke router menuju host tersebut.

3. Jika tidak terdapat entri host yang cocok pada tabel routing, gunakan alamat tujuan datagram yang telah di-mask pada langkah 1 untuk mencari kesamaan di tabel routing. Periksa apakah ada network/subnetwork di tabel routing yang sama dengan alamat network tujuan datagram. Jika ada entri yang sama, kirim frame ke router menuju network/subnetwork tersebut.

4. Jika tidak terdapat entri host ataupun entri network/subnetwork yang sesuai dengan tujuan datagram, host mengirimkan frame ke router default dan menyerahkan proses routing selanjutnya kepada router default.

5. Jika tidak terdapat rute default di tabel routing, semua host diasumsikan dalam keadaan terhubung langsung. Dengan demikian host pengirim akan mencari alamat fisik host tujuan menggunakan ARP.

5.4. Membentuk Tabel Routing

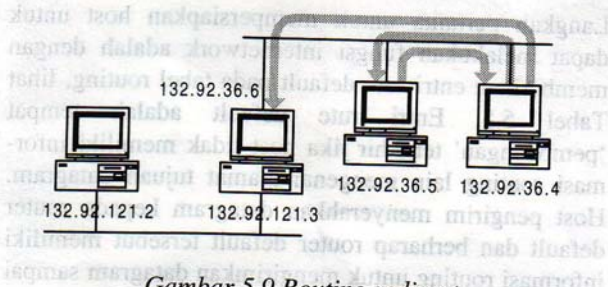
Ketika sebuah host baru dinyalakan, ia belum memiliki cache ARP yang lengkap. Entri pada cache ARP yang dimilikinya hanya untuk host itu sendiri. Setelah berinteraksi dengan host lain barulah host tersebut memiliki entri-entri tambahan pada cache ARP. Hal yang sama juga terjadi pada tabel routing di host. Pada saat host baru dinyalakan, host tersebut tidak memiliki informasi di tabel routing kecuali entri untuk jaringan lokalnya. Tabel routing seperti ini kadang disebut sebagai tabel routing minimal. Dalam kondisi hanya memiliki tabel routing minimal, host belum siap untuk melakukan internetwork karena hanya dapat berkomunikasi dengan host-host yang terletak pada satu jaringan lokal.

Gambar 5.8 Host dengan tabel routing minimal.

Perhatikan bagaimana jika host ini mengirimkan data ke host dengan alamat network yang berbeda. Jika hal tersebut terjadi, tidak ada entri di tabel routing yang menunjukkan ke mana host harus mengirimkan frame. Akibatnya, host akan mencari alamat fisik host tujuan dan tidak akan menemukannya.

Langkah pertama untuk mempersiapkan host untuk dapat melakukan fungsi internetwork adalah dengan memberikan entri rute default pada tabel routing, lihat Tabel 5.5. Entri rute default adalah tempat 'pembuangan' terakhir jika host tidak memiliki informasi routing lain mengenai alamat tujuan datagram. Host pengirim menyerahkan datagram kepada router default dan berharap router default tersebut memiliki informasi routing untuk mengirimkan datagram sampai ke tujuannya.

Tabel 5.5 Tabel routing dengan rute default

IP address	Langsung/tidak	Router	No. interface
132.92.36	langsung	<kosong>	1
0.0.0.0	tak langsung	132.92.36.6	1

Router (dalam hal ini router default) dapat menyatakan bahwa dirinya bukan rute terbaik untuk mencapai host tertentu, melainkan harus melalui router yang lain dalam jaringan lokal berdasarkan tabel routing yang dimilikinya. Jika demikian, maka router tersebut mengirimkan pesan kepada host pengirim datagram menggunakan ICMP redirect dan memberitahukan host pengirim tersebut agar datagram menuju host tertentu dialihkan melalui router lain. Selain itu router meneruskan datagram yang diterimanya ke router lain tersebut sehingga host pengirim tidak perlu mengirim ulang datagram tersebut. Host pengirim menerima pesan ICMP redirect itu dan menambahkan entri host pada tabel routing dengan informasi router yang baru. Proses seperti ini disebut sebagai *routing redirect*. Dengan adanya entri routing yang baru, host pengirim tidak akan mengirim datagram untuk host tadi melalui router default.

Gambar 5.9 Routing redirect

Gambar 5.9 memperlihatkan proses *routing redirect*. Misal tabel routing pada host 132.92.36.5 adalah seperti Tabel 5.6. Dari tabel routing kita ketahui bahwa rute default host tersebut adalah 132.92.36.4. Host 132.92.36.5 hendak mengirim data ke 132.92.121.2. Karena pada tabel tidak ada rute yang menuju host 132.92.121.2, maka frame dikirim ke 132.92.36.4.

Tabel 5.6 Tabel routing host 132.92.36.5

IP address	Langsung/tidak	Router	No. interface
132.92.36	langsung	<kosong>	1
0.0.0.0	tak langsung	132.92.36.4	1

Sementara itu tabel routing pada host 132.92.36.4 yang juga dapat meneruskan datagram (router) adalah seperti pada Tabel 5.7. Pada tabel tersebut terdapat entri rute menuju 132.92.121 yang melalui 132.92.36.6. Ketika router 132.92.36.4 menerima frame dari 132.92.36.5 dan memeriksa tujuan datagram, router melihat bahwa datagram ditujukan kepada 132.92.121.1 dan bukan untuk host itu sendiri. Dengan demikian router perlu menyampaikan datagram tersebut lebih lanjut. Dari hasil pemeriksaan terhadap tabel routing, router

mengetahui bahwa datagram perlu dilewatkan ke router 132.92.36.6 untuk mencapai 132.92.121.1. Karena frame akan dikeluarkan melalui interface yang sama maka router 132.92.36.4 perlu memberitahu host 132.92.36.5 bahwa ia bukan rute terbaik menuju 132.92.121.1, melainkan melalui 132.92.36.6.

Tabel 5.7 Tabel routing router 132.92.36.4

IP address	Langsung/tidak	Router	No. interface
132.92.36	Langsung	<kosong>	1
132.92.121	Tak langsung	132.92.36.6	1
0.0.0.0	Tak langsung	132.92.36.6	1

Router 132.92.36.4 kemudian melakukan dua hal: memberikan pesan ICMP *redirect* kepada 132.92.36.5 dan menyampaikan datagram melalui 132.92.36.6. Host 132.92.36.5 menerima pesan ICMP *redirect* tersebut lalu menambahkan entri host 132.92.121.1 di tabel routing dan menyatakan bahwa untuk mencapai host tersebut perlu melalui router 132.92.36.6. Akibat ICMP *redirect* ini routing tabel host 132.92.36.5 menjadi seperti pada Tabel 5.8

Tabel 5.8 Tabel routing host 132.92.36.5 setelah ICMP redirect

IP address	Langsung/tidak	Router	No. interface
132.92.36	langsung	<kosong>	1
132.92.121.1	tak langsung	132.92.36.6	1
0.0.0.0	tak langsung	132.92.36.6	1

Proses *routing redirect* seperti di atas adalah metode paling sederhana dalam membentuk tabel routing sebuah host. Router hanya boleh memberikan pesan

ICMP *redirect* untuk host, dan tidak untuk subnet. Karena alasan tersebut metode ini menyebabkan tabel routing hanya terisi dengan entri-entri host yang dihubungi tidak melalui rute default. Metode ini paling sesuai diterapkan pada host-host yang terletak di atau yang dekat dengan titik ujung jaringan TCP/IP. Metode ini tidak sesuai jika diterapkan pada host-host yang terletak pada jaringan tulang punggung, apalagi jika digunakan pada router-router jaringan TCP/IP.

Metode lain yang dapat digunakan untuk membentuk tabel routing adalah dengan metode routing statik. Pada metode ini entri-entri rute di host dan router diisikan secara manual. Metode ini lebih bagus daripada hanya memanfaatkan ICMP *redirect* karena tabel routing dapat diisi dengan IP address jaringan dan tidak terbatas pada IP address host saja. Pembuatan tabel routing secara manual umumnya lebih disukai dalam jaringan TCP/IP yang terdiri atas hanya beberapa router karena jumlah entri dan jumlah router yang harus dikonfigurasi hanya sedikit. Semakin besar jaringan TCP/IP (dihitung dari jumlah router) semakin banyak usaha yang diperlukan untuk membuat tabel routing yang lengkap di tiap-tiap router.

Hitungan kasar jumlah total entri tabel routing yang perlu dimasukkan di seluruh router adalah berbanding pangkat dua terhadap jumlah router yang ada di jaringan tersebut. Jumlah yang besar ini dapat menyebabkan ketidakkonsistenan dalam pengisian entri tabel routing. Dari kenyataan di atas kita melihat ada suatu batasan ukuran jaringan (yang besarnya berbeda untuk setiap orang) ketika pengisian tabel routing secara statik menjadi sulit dilakukan karena sedemikian banyaknya entri yang harus dimasukkan. Di samping

kelemahan tersebut, pengisian tabel routing secara manual tidak mencerminkan dinamika jaringan.

Selain kedua metode di atas, pembentukan tabel routing dapat dilakukan dengan menggunakan protokol routing. Protokol routing adalah protokol yang digunakan oleh router-router untuk saling bertukar informasi routing. Router-router pada jaringan TCP/IP membentuk tabel routing berdasarkan informasi routing yang dipertukarkan setiap selang waktu tertentu. Pertukaran informasi antar router dapat menunjukkan dinamika jaringan karena jika pada suatu saat ada router atau jaringan yang putus berarti router-router yang lain tidak akan menerima informasi routing dari router yang putus tersebut. Jika hal ini terjadi, router-router dalam jaringan akan menghapus entri tabel routing yang informasinya tidak diperbarui. Karena penggunaan protokol routing dapat mencerminkan dinamika jaringan maka metode ini disebut juga dengan routing dinamik.

Keunggulan penggunaan protokol routing yang lain adalah fleksibilitas dan konfigurasi yang umumnya relatif sederhana untuk jaringan yang besar. Kita dapat memilih menggunakan protokol routing yang akan digunakan di jaringan TCP/IP sesuai dengan karakteristik protokol routing yang diinginkan. Keunggulan-keunggulan inilah yang menyebabkan mengapa routing dinamik menjadi pilihan untuk membentuk tabel routing pada jaringan TCP/IP.

5.5. Protokol Routing

Protokol routing yang umum digunakan pada jaringan TCP/IP saat ini adalah Routing Information Protocol (RIP), Open Shortest Path First (OSPF), dan Border Gateway Protocol (BGP). Protokol-protokol tersebut dimasukkan dalam kategori yang berbeda. RIP dan OSPF termasuk kategori *interior gateway protocol* (IGP) sedangkan BGP termasuk kategori *exterior gateway protocol* (EGP). IGP adalah protokol yang menangani routing jaringan internet pada sebuah *autonomous system* sementara EGP menangani routing antar *autonomous system*.

Perlunya penggunaan *exterior gateway protocol* didasarkan pada kenyataan bahwa *interior gateway protocol* tidak dirancang untuk jaringan yang sangat besar. Akibatnya, routing jaringan TCP/IP yang terdiri atas banyak jaringan perlu menggunakan hierarki dengan membagi jaringan tersebut menjadi kumpulan autonomous system. Autonomous system (AS) secara umum didefinisikan sebagai jaringan internet yang berada dalam satu kendali administrasi dan teknis. Internet yang begitu besar sekarang ini adalah kumpulan dari ribuan autonomous system.

Exterior gateway protocol memiliki kemampuan *policy routing* karena sebagian *autonomous system* di Internet mempunyai kebijakan dalam hal routing. Salah satu contoh yang jelas adalah NSFnet. Pada perkembangan Internet sampai dengan tahun 1995 NSFnet menjadi bagian dari jaringan tulang punggung Internet. Ini berarti NSFnet bersedia melewatkan bersedia melewatkan datagram dari dan ke seluruh bagian Internet. Pada tahun 1995 NSFnet memutuskan untuk kembali

menjadi jaringan riset dan mengambil kebijakan untuk tidak lagi bersedia menjalankan fungsi seperti dahulu. NSFnet sekarang hanya bersedia melewatkan datagram dari atau ke jaringan-jaringan universitas dan lembaga riset yang bergabung dengannya.

Buku ini tidak membahas mengenai exterior gateway protocol karena umumnya hanya jaringan yang hendak menetapkan policy routing yang perlu menggunakan jenis protokol ini. Kita akan memfokuskan pembicaraan kepada interior gateway protocol yang banyak digunakan di jaringan TCP/IP yaitu RIP dan OSPF.

Sebelum kita membahas kedua protokol routing di atas secara lebih detail, kita perhatikan karakteristik kedua protokol routing. Karakteristik kedua protokol tersebut berbeda dan ini menjadi dasar pemilihan penggunaan protokol routing di jaringan.

Karakteristik Routing Information Protocol (RIP) adalah sebagai berikut:

- Menggunakan algoritma *distance-vector* (Bellman-Ford)
- Dapat menyebabkan *routing loop*
- Diameter jaringan terbatas
- Lambat mengetahui perubahan jaringan
- Menggunakan metrik tunggal

Karakteristik Open Shortest Path First (OSPF)

- Menggunakan algoritma *link-state*
- Membutuhkan waktu CPU dan memori yang besar
- Tidak menyebabkan *routing loop*

- Dapat membentuk hierarki routing menggunakan konsep area
- Cepat mengetahui perubahan pada jaringan
- Dapat menggunakan beberapa macam metrik

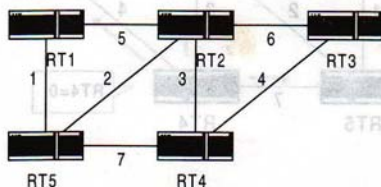
5.6. Routing Information Protocol

RIP adalah protokol routing yang menggunakan algoritma *distance-vector* atau yang juga dikenal sebagai algoritma Bellman-Ford. Protokol RIP sudah ada sejak masa ARPANET dan pemakaiannya di Intenet diadopsi dari protokol jaringan XNS dari Xerox. Protokol RIP cukup sederhana dan mudah diimplementasikan dalam jaringan TCP/IP. Kelemahan algoritma *distance-vector* adalah lambat dalam mengetahui perubahan jaringan dan dapat menimbulkan *routing loop*.

Routing loop adalah suatu kondisi ketika dua router bertetangga saling mengira bahwa untuk mencapai suatu alamat, datagram seharusnya dilewatkan ke router tetangganya tersebut. Misal: router RT1 menyangka bahwa untuk mencapai RT3 datagram harus dilewatkan melalui router RT2 tetapi router RT2 menyangka untuk mencapai RT3 harus melalui router RT1. *Routing loop* dapat juga terjadi antara tiga router, yaitu router RT1 menyampaikan datagram ke host D melalui router RT2, router RT2 menyampaikan datagram tersebut melalui router RT3, dan router RT3 menyampaikan datagram tersebut melalui router RT1. Jika *routing loop* seperti ini terjadi, datagram akan berputar-putar (*looping*) antara router-router tersebut sampai TTL untuk datagram tersebut habis.

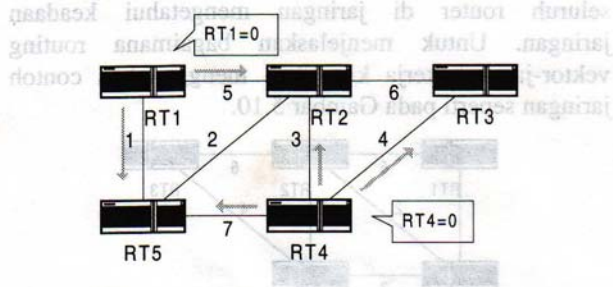
Bagian ini akan menjelaskan prinsip kerja routing vektor-jarak, bagaimana *routing loop* dapat terjadi dengan routing seperti ini, cara-cara yang digunakan untuk memperkecil kemungkinan *loop* tersebut, dan juga mengenai protokol RIP versi 1 dan versi 2 yang saat ini sudah terdokumentasi dalam RFC.

5.6.1. Routing vektor-jarak

Telah disebutkan di atas bahwa algoritma routing *distance-vector* cukup sederhana. Algoritma ini membentuk tabel routing di jaringan adalah dengan cara setiap router memberikan informasi mengenai keadaan jaringan yang diketahui router tersebut kepada router-router tetangganya setiap selang waktu tertentu. Informasi keadaan jaringan tersebut adalah dalam bentuk *distance-vector* (vektor-jarak), yaitu jumlah hop yang diperlukan untuk mencapai suatu jaringan. Router tetangga tersebut menyimpan dan mengolah informasi keadaan jaringan yang diterimanya dan juga menyampaikan informasi yang dimilikinya ke router-router tetangga yang lain. Hal ini terus berlangsung sampai seluruh router di jaringan mengetahui keadaan jaringan. Untuk menjelaskan bagaimana routing vektor-jarak bekerja kita akan menggunakan contoh jaringan seperti pada Gambar 5.10.

Gambar 5.10 Jaringan komputer dengan 5 router

Kita asumsikan bahwa semua router di jaringan dalam keadaan baru dinyalakan. Pada saat ini semua router tidak memiliki informasi vektor-jarak kecuali kepada dirinya sendiri. Informasi vektor-jarak tersebut disimpan dalam bentuk tabel routing. Jadi pada saat awal, tabel routing masing-masing router mirip dengan tabel routing RIP router RT1, yaitu:

Dari RT1 ke	Jalur	Hop
RT1	lokal	0

Setelah router mulai menjalankan algoritma vektor-jarak, router-router mulai memberikan informasi vektor-jarak ke router-router tetangganya. Untuk memudahkan penjelasan, kita asumsikan bahwa router RT1 paling dulu mengirimkan informasi vektor-jarak ke router-router tetangganya, RT2 dan RT5. Jadi, pada saat ini router RT1 mengirimkan vektor-jarak ke jalur 1 dan 5. Informasi yang dikirim kira-kira berbunyi "jarak saya ke RT1 adalah 0 hop". Dalam waktu yang hampir berdekatan router RT4 juga mengirimkan vektor-jarak ke jalur 3, 4 dan 7.

Gambar 5.11 Router RT1 dan RT4 mengirim informasi vektor-jarak ke tetangganya

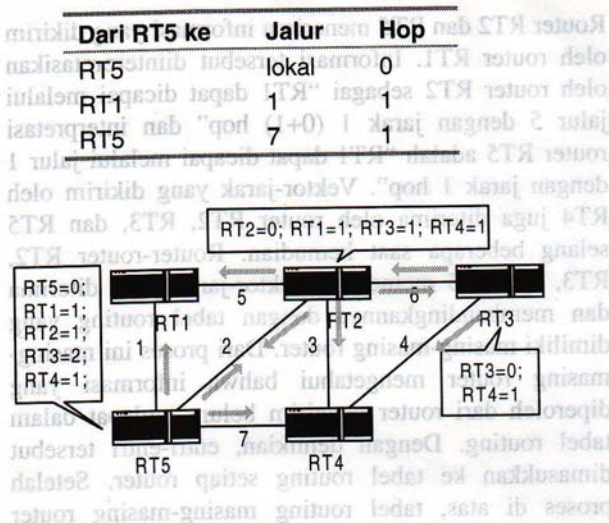
Router RT2 dan RT5 menerima informasi yang dikirim oleh router RT1. Informasi tersebut diinterpretasikan oleh router RT2 sebagai "RT1 dapat dicapai melalui jalur 5 dengan jarak 1 (0+1) hop" dan interpretasi router RT5 adalah "RT1 dapat dicapai melalui jalur 1 dengan jarak 1 hop". Vektor-jarak yang dikirim oleh RT4 juga diterima oleh router RT2, RT3, dan RT5 selang beberapa saat kemudian. Router-router RT2, RT3, dan RT5 memeriksa vektor-jarak yang diterima dan membandingkannya dengan tabel routing yang dimiliki masing-masing router. Dari proses ini masing-masing router mengetahui bahwa informasi yang diperoleh dari router pengirim belum terdapat dalam tabel routing. Dengan demikian, entri-entri tersebut dimasukkan ke tabel routing setiap router. Setelah proses di atas, tabel routing masing-masing router adalah seperti di bawah ini.

Dari RT1 ke	Jalur	Hop
RT1	lokal	0

Dari RT2 ke	Jalur	Hop
RT2	lokal	0
RT1	5	1
RT4	3	1

Dari RT3 ke	Jalur	Hop
RT3	lokal	0
RT4	4	1

Dari RT4 ke	Jalur	Hop
RT4	lokal	0

Dari RT5 ke	Jalur	Hop
RT5	lokal	0
RT1	1	1
RT5	7	1

Gambar 5.12 Router RT2, RT3, dan RT5 mengirim informasi vektor-jarak ke tetangganya

Pada giliran berikutnya, router RT3, RT2, dan RT5 berturut-turut dalam waktu yang hampir bersamaan mengirimkan vektor-jarak ke tetangga masing-masing. Pada saat ini kita asumsikan bahwa router RT2 lebih dahulu menerima vektor-jarak dari router RT3 sebelum sempat mengirimkan vektor-jarak. Router RT2 memeriksa informasi dari RT3 tersebut lalu membandingkannya dengan entri tabel routing yang sudah ada. Hasil pemeriksaan menunjukkan entri RT3 belum ada di tabel routing, sehingga entri ini dimasukkan ke dalam tabel routing router RT2. Dari vektor-jarak yang dikirim RT3, router RT2 mengetahui bahwa ia dapat mencapai RT4 melalui jalur 6 dengan jarak 2 hop. Pada saat yang sama di tabel routing terdapat entri router RT4 yang dapat dicapai melalui jalur 3 dan

hanya berjarak 1 hop. Ini berarti jarak dari RT2 ke RT4 melalui jalur 3 lebih pendek daripada melalui jalur 6. Dengan demikian, informasi vektor-jarak untuk RT4 dari RT3 tidak digunakan oleh router RT2. Pada saat yang hampir bersamaan pula router RT4 menerima vektor-jarak dari RT3 dan memperbarui tabel routing yang dimilikinya. Tabel routing router RT2 dan RT4 setelah RT3 mengirim vektor-jarak adalah seperti di bawah ini.

Dari RT2 ke	Jalur	Hop
RT2	lokal	0
RT1	5	1
RT3	6	1
RT4	3	1

Dari RT4 ke	Jalur	Hop
RT4	lokal	0
RT3	4	1

Router RT2 mengirimkan vektor-jarak berdasarkan tabel routing yang baru ke semua jalur yang terhubung dengannya. Router-router yang menerima vektor-jarak dari RT2, termasuk router RT5, kemudian memperbarui tabel routing masing-masing menggunakan algoritma vektor-jarak. Sekejap kemudian router RT5 mengirimkan vektor-jarak berdasarkan tabel routing terbaru yang dimilikinya. Setelah router RT5 mengirim vektor-jarak, tabel routing di jaringan menjadi seperti di bawah ini.

Dari RT1 ke	Jalur	Hop
RT1	lokal	0
RT2	5	1
RT3	5	2
RT4	5	2
RT5	1	1

Dari RT2 ke	Jalur	Hop
RT2	lokal	0
RT1	5	1
RT3	6	1
RT4	3	1
RT5	2	1

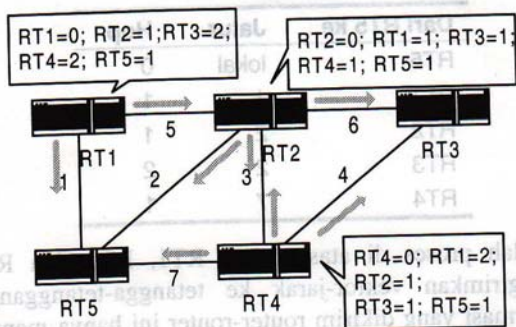
Dari RT3 ke	Jalur	Hop
RT3	lokal	0
RT1	6	2
RT2	6	1
RT4	4	1

Dari RT4 ke	Jalur	Hop
RT4	lokal	0
RT1	3	2
RT2	3	1
RT3	4	1
RT5	7	1

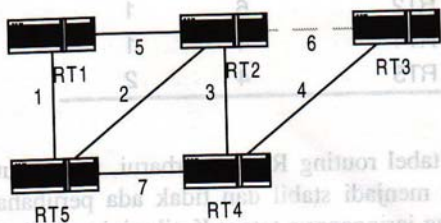
Dari RT5 ke	Jalur	Hop
RT5	lokal	0
RT1	1	1
RT2	2	1
RT3	2	2
RT4	7	1

Setelah proses di atas, router RT1, RT2 dan RT4 mengirimkan vektor-jarak ke tetangga-tetangganya. Informasi yang dikirim router-router ini hanya menyebabkan perubahan tabel routing di router RT3. Kita asumsikan informasi dari RT4 lebih dahulu diterima daripada dari RT2 sehingga rute ke RT5 dari RT3 melalui jalur 4 dengan jarak 2 hop. Ketika informasi dari RT2 diterima, jarak dari RT3 ke RT5 melalui jalur 6 juga berjarak 2 hop. Jarak ini sama dengan yang terdapat di tabel routing router RT3 sehingga informasi dari RT2 tidak dimasukkan ke dalam tabel routing RT3. Tabel routing RT3 sekarang terlihat seperti tabel di bawah.

Dari RT3 ke	Jalur	Hop
RT3	lokal	0
RT1	6	2
RT2	6	1
RT4	4	1
RT5	4	2

Setelah tabel routing RT3 diperbarui, tabel routing di jaringan menjadi stabil dan tidak ada perubahan lagi sepanjang jaringannya tetap. Ketika ini terjadi, routing disebut sudah konvergen.

Gambar 5.13 Router RT1, RT2, dan RT4 mengirim informasi vektor-jarak ke tetangganya

5.6.2. Perubahan kondisi jaringan

Kondisi jaringan tidak akan stabil untuk seterusnya, terkadang ada jalur yang putus. Penyebab putusnya pun macam-macam, mulai dari kabel yang digigit tikus sampai yang terkena cangkul pada saat proyek pembangunan fisik. Apa pun penyebabnya, jalur yang putus menyebabkan kondisi jaringan berubah dan akan tampak dalam tabel routing.

Gambar 5.14 Jaringan berubah akibat jalur 6 terputus

Kita coba perhatikan apa yang terjadi jika jalur 6 terputus. Setelah jalur 6 terputus, router RT2 dan RT3 segera mengetahuinya. Bagi kedua router, semua entri di tabel routing yang menggunakan jalur 6 tidak dapat lagi digunakan dan hop untuk jalur itu diberi nilai tak-hingga. Tabel routing yang baru untuk router RT2 dan RT3 adalah seperti di bawah ini.

Dari RT2 ke	Jalur	Hop
RT2	lokal	0
RT1	5	1
RT3	6	∞
RT4	3	1
RT5	2	1

Dari RT3 ke	Jalur	Hop
RT3	lokal	0
RT1	6	∞
RT2	6	∞
RT4	4	1
RT5	4	2

Perubahan ini memicu penyebaran vektor-jarak yang baru dalam jaringan yang berasal dari router RT2 dan RT3. RT1 misalnya, mengetahui dari RT2 bahwa nilai hop RT3 sekarang bernilai tak-hingga. Karena entri RT3 di tabel routing sebelumnya melalui jalur 2, maka nilai hop untuk RT3 menjadi tak-hingga. Tabel routing di RT1 dan RT5 berubah setelah menerima informasi baru sementara tabel routing di RT4 tidak berubah.

Dari RT1 ke	Jalur	Hop
RT1	lokal	0
RT2	5	1
RT3	5	∞
RT4	5	2
RT5	1	1

Dari RT5 ke	Jalur	Hop
RT5	lokal	0
RT1	1	1
RT2	2	1
RT3	2	∞
RT4	7	1

Setelah perubahan tersebut, berturut-turut router RT1, RT5, dan RT4 mengirimkan vektor-jarak ke seluruh jalur yang terhubung langsung dengannya. Informasi vektor-jarak yang terbaru ini mengubah tabel routing di RT2, RT3, dan RT5. Router RT3 yang sebelumnya menganggap jalur ke RT1 dan RT2 terputus sekarang dapat mencapai keduanya melalui jalur 4 dengan jarak 2 hop. Demikian pula di RT5, jalur menuju RT3 yang tidak dapat dicapai melalui jalur 2 sekarang dialihkan melalui jalur 7 dengan jarak 2 hop. Tabel routing router RT2, RT3, dan RT5 sekarang menjadi berikut.

Dari RT2 ke	Jalur	Hop
RT2	lokal	0
RT1	5	1
RT3	3	2
RT4	3	1
RT5	2	1

Dari RT3 ke	Jalur	Hop
RT3	lokal	0
RT1	4	3
RT2	4	2
RT4	4	1
RT5	4	2

Dari RT5 ke	Jalur	Hop
RT5	lokal	0
RT1	1	1
RT2	2	1
RT3	7	2
RT4	7	1

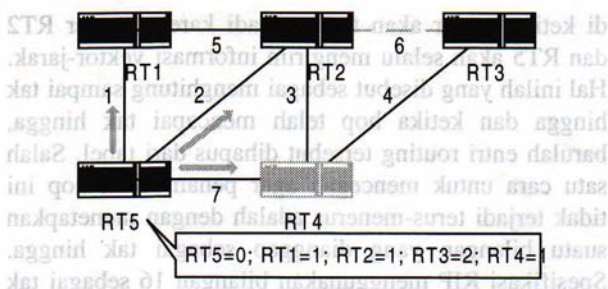
Penyebaran informasi vektor-jarak berikutnya adalah oleh router RT2 dan RT3. Hasil dari penyebaran vektor-jarak ini adalah berubahnya tabel routing di RT1. Router RT1 sekarang mengetahui bahwa untuk mencapai RT3 tetap melalui jalur 5 tetapi dengan jarak 3 hop. Setelah perubahan tabel routing di RT1, routing kembali konvergen dan mencerminkan kondisi terakhir jaringan.

Dari RT1 ke	Jalur	Hop
RT1	lokal	0
RT2	5	1
RT3	5	3
RT4	5	2
RT5	1	1

5.6.3. Menghitung sampai tak-hingga

Salah satu kelemahan algoritma routing vektor-jarak adalah menghitung sampai tak-hingga (*counting to infinity*). Untuk mengetahui bagaimana hal ini dapat terjadi, perhatikan kembali Gambar 5.14. Di gambar tersebut terlihat bahwa jalur 6 terputus dan menyebabkan perubahan kondisi jaringan. Misalnya kemudian masalah lain muncul pada jaringan, yaitu router RT4 mati. Router-router RT2 dan RT5 baru menganggap router RT4 mati ketika dalam selang waktu tertentu router-router ini tidak menerima informasi vektor-jarak dari RT4. Menghitung sampai tak-hingga menjadi muncul dalam kasus ini karena ketika router RT2 menganggap router RT4 mati, router RT5 belum menganggap RT4 mati atau sebaliknya.

Pada jaringan di atas dapat terjadi router RT2 telah menganggap router RT4 mati dan menghapus entri RT4 dari tabel routing sementara router RT5 masih belum menganggap RT4 mati. Sesaat setelah router RT2 menganggap RT4 mati, router RT5 mengirimkan vektor-jarak. Router RT2 yang menerima informasi itu kembali memasukkan RT4 ke tabel routing. Entri RT4 dalam tabel routing RT1, RT2, dan RT5 saat ini adalah seperti berikut:

Entri RT4 di router	Jalur	Hop
RT1	5	2
RT2	2	2
RT5	7	1

Gambar 5.15 Router RT5 mengirim vektor-jarak ketika router RT4 dianggap mati oleh router RT2

Sebelum router RT2 sempat mengirimkan vektor-jarak yang dimilikinya, router RT5 telah menganggap RT4 mati dan menghapus entri RT4 dari tabel routing. Jadi, ketika RT2 mengirim vektor-jarak, RT5 memasukkan entri RT4 ke tabel routing. Entri RT4 dalam tabel routing kemudian menjadi berikut:

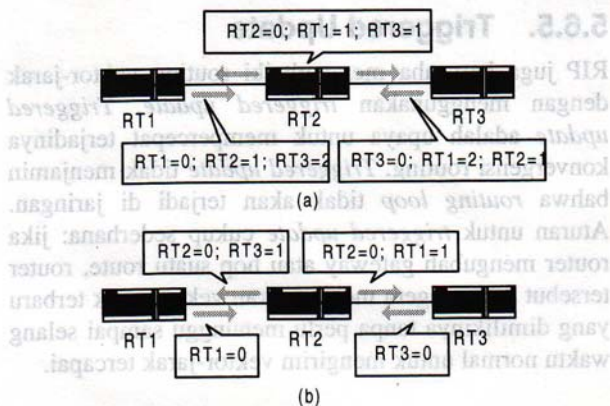
Entri RT4 di router	Jalur	Hop
RT1	5	3
RT2	2	2
RT5	2	3

Saat ini antara router RT2 dan RT5 telah terbentuk routing loop. Terlihat bahwa router RT2 mempercayai bahwa untuk mengirim datagram ke RT4 perlu melalui RT5, sementara router RT5 mempercayai yang sebaliknya. Jika router RT5 mengirim vektor-jarak, hop dari RT2 ke RT4 bertambah satu tetapi hop dari RT1 ke RT4 tidak berubah. Jika router RT2 mengirim vektor-jarak, hop dari RT1 ke RT4 dan RT5 ke RT4 bertambah satu. Pertambahan jumlah hop menuju RT4

di ketiga router akan terus terjadi karena router RT2 dan RT5 akan selalu mengirim informasi vektor-jarak. Hal inilah yang disebut sebagai menghitung sampai tak hingga dan ketika hop telah mencapai tak hingga, barulah entri routing tersebut dihapus dari tabel. Salah satu cara untuk mencegah agar penambahan hop ini tidak terjadi terus-menerus adalah dengan menetapkan suatu bilangan yang dianggap sebagai tak hingga. Spesifikasi RIP menggunakan bilangan 16 sebagai tak hingga, dan ini berarti diameter jaringan (jumlah hop maksimum dalam jaringan) yang menggunakan RIP tidak dapat lebih dari 16.

5.6.4. Split Horizon

Split Horizon adalah suatu cara RIP untuk memperbaiki kinerja protokol routing tersebut. Dengan *split horizon*, RIP memperkecil kemungkinan terjadinya *routing loop* dan menghitung sampai tak hingga dalam jaringan. Prinsip *split horizon* sangat sederhana: router tidak perlu mengirim suatu vektor-jarak ke jalur tempat ia menerima informasi vektor-jarak tersebut.

Pada Gambar 5.16 terlihat tiga buah router: RT1, RT2, dan RT3, yang menjalankan routing vektor-jarak. Jika menggunakan vektor-jarak biasa, maka informasi yang dikirim ke jaringan adalah seperti pada Gambar 5.16-a, sedangkan jika menggunakan *split horizon*, akan terlihat seperti pada Gambar 5.16-b.

Gambar 5.16 Routing vektor-jarak: a. tanpa *split horizon*; b. dengan *split horizon*

Split horizon terdiri atas dua jenis: *split horizon* normal dan split horizon dengan *poisonous reverse*. *Split horizon* normal adalah seperti yang terlihat pada Gambar 5.16-b. *Split horizon* dengan *poisonous reverse* lebih baik daripada *split horizon* biasa karena tetap mengirim suatu informasi vektor-jarak ke jalur tempat ia menerima vektor-jarak tersebut dengan membuat nilai vektor-jarak tersebut menjadi tak-hingga. Membuat nilai vektor-jarak menjadi tak-hingga, alih-alih tidak mengirimkan vektor-jarak, menciptakan semacam jawaban positif kepada jaringan. Dengan jawaban positif ini, router-router mengetahui bahwa memang terdapat jaringan yang nilai vektor-jaraknya "diracuni" menjadi tak-hingga. Jika hanya menggunakan *split horizon* biasa, router-router tidak dapat mengetahui apakah suatu jaringan memang ada karena tidak ada informasi vektor-jarak yang diterima.

5.6.5. Triggered Update

RIP juga berusaha memperbaiki routing vektor-jarak dengan menggunakan *triggered update*. *Triggered update* adalah upaya untuk mempercepat terjadinya konvergensi routing. *Triggered update* tidak menjamin bahwa *routing loop* tidak akan terjadi di jaringan. Aturan untuk *triggered update* cukup sederhana: jika router mengubah gateway atau hop suatu route, router tersebut harus segera mengirimkan vektor-jarak terbaru yang dimilikinya tanpa perlu menunggu sampai selang waktu normal untuk mengirim vektor-jarak tercapai.

5.7. RIP versi 1

Spesifikasi protokol routing ini ditulis dalam dokumen RFC1058. RIP merupakan routing vektor-jarak yang dimodifikasi dengan *triggered update* dan *split horizon* dengan *poisonous reverse* untuk meningkatkan kinerjanya. RIP ditujukan agar host dan router dapat bertukar informasi untuk menghitung rute dalam jaringan TCP/IP. Informasi yang dipertukarkan RIP adalah informasi mengenai 'tujuan' yang dapat berupa host, network, subnet, atau rute default.

Setiap host yang menjalankan RIP versi 1 memiliki tabel routing yang setidaknya berisi:

- IP address tujuan (host, subnet, network, atau rute default).
- Metrik yang menunjukkan 'biaya' total untuk mencapai tujuan tersebut dari host.
- IP address router yang perlu dilalui.

- Sebuah tanda untuk menunjukkan apakah rute baru saja berubah.
- Beberapa timer.

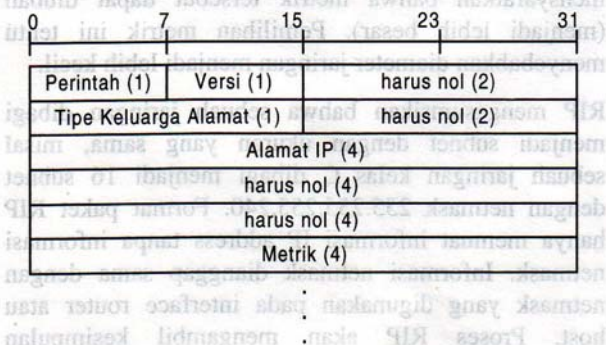
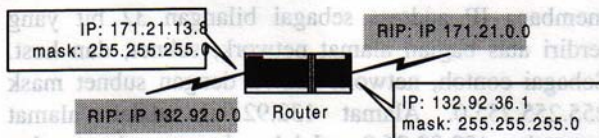
Metrik yang digunakan RIP untuk menentukan rute sangat sederhana, yaitu adalah 'jumlah hop' antara router dengan tujuan. Metrik ini adalah bilangan bulat 1 sampai dengan 15, dan 16 dianggap sebagai tak-hingga. Pembatasan metrik ini adalah konsekuensi penggunaan routing vektor-jarak dan berakibat membatasi diameter jaringan. Metrik interface jaringan secara default adalah 1 walaupun spesifikasi RIP mensyaratkan bahwa metrik tersebut dapat diubah (menjadi lebih besar). Pemilihan metrik ini tentu menyebabkan diameter jaringan menjadi lebih kecil.

RIP mengasumsikan bahwa sebuah jaringan dibagi menjadi subnet dengan ukuran yang sama, misal sebuah jaringan kelas C dibagi menjadi 16 subnet dengan netmask 255.255.255.240. Format paket RIP hanya memuat informasi IP address tanpa informasi netmask. Informasi netmask dianggap sama dengan netmask yang digunakan pada interface router atau host. Proses RIP akan mengambil kesimpulan berdasarkan IP address tersebut saja untuk menentukan apakah IP address yang terdapat di paket RIP termasuk IP address network, subnet, host atau default route. RIP membaca IP address sebagai bilangan 32 bit yang terdiri atas bagian alamat network, subnet, dan host. Sebagai contoh, network 132.92 dengan subnet mask 255.255.255.0. Alamat 132.92.0.0 adalah alamat network, 132.92.36.0 adalah alamat subnet, dan 132.92.36.21 adalah alamat host. RIP memeriksa apakah bagian host dari sebuah IP address tidak terdiri

atas angka 0. Jika demikian, RIP langsung mengambil kesimpulan bahwa alamat tersebut merupakan alamat host. RIP mencegah informasi routing alamat subnet keluar dari networknya. Jika terdapat router RIP yang berbatasan dengan network lain, RIP hanya membuat satu entri saja yaitu entri alamat network. Sebagai contoh: sebuah router dengan dua interface dengan nomor network yang berbeda, 132.92.36.22 subnetmask 255.255.255.0 dan 171.21.13.8 subnetmask 255.255.255.0. RIP hanya mengirimkan satu entri 132.92 ke interface 171.21.13.8 dan juga satu entri 171.21 ke interface 132.92.36.22.

0 7 15 23 31

0	7	15 23 31
Perintah (1)	Versi (1)	harus nol (2)
Tipe Keluarga Alamat (1)		harus nol (2)
Alamat IP (4)		
harus nol (4)		
harus nol (4)		
Metrik (4)		

:
:

Gambar 5.17 Router RIP pada perbatasan network

Gambar 5.18 Format Paket RIP versi 1

Gambar 5.18 memperlihatkan format paket RIP (versi 1). Pada gambar tersebut hanya diperlihatkan data untuk satu entri saja. Panjang maksimum paket RIP adalah 512 byte, tanpa header UDP. Entri vektor-jarak RIP dimulai dari tipe keluarga alamat sampai metrik. Entri ini dapat muncul sampai 25 kali. Jenis perintah (*command*) pada paket RIP yang digunakan adalah *request* dan *response*. Perintah *request* adalah untuk meminta router mengirimkan tabel routing sementara perintah *response* adalah untuk mengirimkan tabel routing. Versi RIP dalam paket tersebut adalah 1 dan tipe keluarga alamat (*address family identifier*) untuk IP adalah 2. IP address dalam paket adalah alamat biasa dan field metrik berisi bilangan bulat dari 1 sampai 15 untuk entri yang aktif. Metrik 16 digunakan untuk entri yang tidak dapat dicapai.

Pertukaran informasi routing dilakukan RIP setiap 30 detik. Pertukaran informasi routing tersebut pada jaringan lokal dengan beberapa router dapat mengakibatkan kongesti (*congestion*). Kongesti dapat terjadi karena router-router mengirimkan paket RIP dalam waktu yang hampir bersamaan. Untuk menghindari kongesti, router dapat menambah waktu pertukaran itu secara acak menjadi sedikit lebih dari 30 detik. Router-router yang menjalankan RIP bertukar informasi dengan cara mem-broadcast paket RIP *response* menggunakan UDP pada port 520. Sementara router di jaringan mem-broadcast paket RIP masing-masing, host-host yang terletak di jaringan yang sama dapat mendengar pesan tersebut. Host-host tersebut juga menjalankan RIP tetapi hanya mendengarkan paket-paket RIP yang disampaikan router dan tidak mengirimkan pesan. Inilah yang disebut sebagai node

diam (*silent node*), yaitu node-node hanya yang mendengar paket RIP *response* dan membentuk tabel routing berdasarkan paket-paket tersebut.

Pada saat terjadi pertukaran informasi RIP, karena *triggered update* ataupun yang reguler, timer untuk entri-entri tabel routing yang aktif diperbarui lagi. Jika sebuah entri tabel routing tidak diperbarui dalam 180 detik, entri tersebut dianggap kadaluwarsa dan siap dihapus dari tabel. Sebuah entri tabel routing juga dapat dihapus jika metrik untuk entri tersebut dibuat menjadi 16 (tak-hingga) oleh router yang perlu dilaluinya. Metrik entri-entri tabel yang hendak dihapus pertama-tama diubah menjadi 16 dan dibiarkan dahulu di tabel routing 120 detik sebelum benar-benar dihapus. Karena metrik entri tersebut berubah, maka secara otomatis router perlu mengirimkan paket RIP karena dipicu oleh perubahan tabel routing (*triggered update*). Spesifikasi RIP menyebutkan bahwa paket RIP yang dikirim karena triggered update hanya perlu berisi entri-entri tabel yang berubah. Selama 120 detik tersebut, entri-entri yang hendak dihapus itu juga disertakan dalam paket RIP *response* yang dikirim setiap 30 detik. Tujuan menyertakan entri-entri tersebut adalah untuk lebih menjamin agar router-router di jaringan juga akan menghapusnya dari tabel routing.

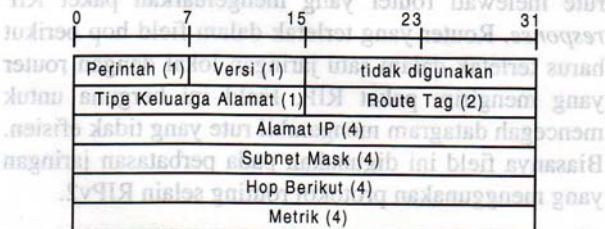
5.8. RIP versi 2

Protokol ini dapat dikatakan sebagai perluasan dari RIP versi 1 dengan menambahkan beberapa kemampuan baru. RIP versi 2 (RIPv2) sama sekali tidak mengubah algoritma routing vektor-jarak yang digunakan RIPv1. Perubahan yang dilakukan RIPv2 hanya

pada format paket RIP yang bersifat menambah informasi yang dikirimkan. Kemampuan-kemampuan baru RIPv2 adalah: *tag* untuk rute eksternal, subnet mask, alamat hop berikut, dan autentikasi.

Perintah tipe keluarga alamat, IP address serta metrik pada RIPv2 memiliki arti yang sama dengan RIPv1. Versi yang digunakan pada paket RIPv2 adalah 2. Dari Gambar 5.19 terlihat bahwa RIPv2 menggunakan bagian dari paket yang pada RIPv1 harus diisi nol.

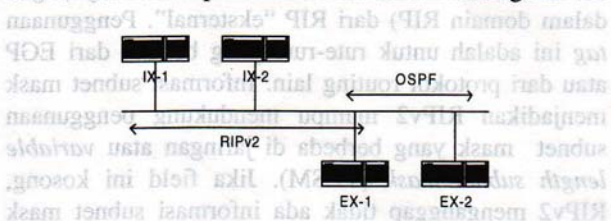
0	7	15	23 31
Perintah (1)	Versi (1)	tidak digunakan	
Tipe Keluarga Alamat (1)		Route Tag (2)	
Alamat IP (4)			
Subnet Mask (4)			
Hop Berikut (4)			
Metrik (4)			

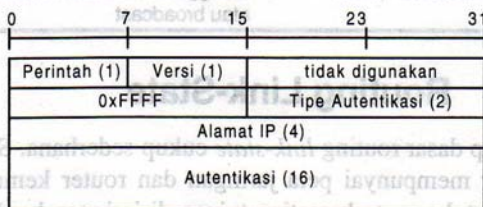
.
.
.

Gambar 5.19 Format Paket RIP versi 2

Tag untuk rute eksternal memberikan kemampuan bagi RIPv2 untuk membedakan RIP "internal" (jaringan dalam domain RIP) dari RIP "eksternal". Penggunaan *tag* ini adalah untuk rute-rute yang berasal dari EGP atau dari protokol routing lain. Informasi subnet mask menjadikan RIPv2 mampu mendukung penggunaan subnet mask yang berbeda di jaringan atau *variable length subnet mask* (VLSM). Jika field ini kosong, RIPv2 menganggap tidak ada informasi subnet mask yang dikirimkan. Interaksi RIPv2 dengan RIPv1 mengenai informasi IP address dan subnet mask mengingat RIPv1 menganggap jaringan subnet mask yang

sama. Cara yang digunakan RIPv2 untuk berinteraksi dengan RIPv1 adalah tidak mengirimkan informasi rute dengan subnet mask yang lebih spesifik karena RIPv1 akan menganggapnya sebagai rute menuju host.

Router yang menjalankan RIPv2 dapat memberi tahu agar router lain tidak melewatkan suatu rute melalui dirinya dengan cara mengisi field alamat hop berikut dengan router yang harus dilalui oleh datagram menuju rute tersebut. Field yang diisi dengan 0.0.0.0 berarti rute melewati router yang mengeluarkan paket RIP *response*. Router yang terletak dalam field hop berikut harus terletak dalam satu jaringan lokal dengan router yang mengirim paket RIP. Field ini berguna untuk mencegah datagram mengambil rute yang tidak efisien. Biasanya field ini digunakan pada perbatasan jaringan yang menggunakan protokol routing selain RIPv2.

Di Gambar 5.20 terlihat empat router yang terhubung pada suatu jaringan lokal. Router IX-1, IX-2, dan EX-1 menjalankan RIPv2. EX-1 dan EX-2 menjalankan protokol OSPF. EX-1 dapat memanfaatkan field alamat hop berikut untuk menunjukkan rute apa saja yang harus dilewatkan melalui router EX-2. Dengan demikian router EX-2 tidak perlu ikut serta dalam routing RIPv2.

Gambar 5.20 Router EX-1 menggunakan field 'alamat hop berikut'

RIPv2 juga dapat menggunakan autentikasi seperti terlihat pada Gambar 5.21. Karena paket RIPv2 dengan autentikasi menghabiskan data setara dengan satu entri route, maka jumlah maksimum entri rute dalam satu paket RIPv2 turun menjadi 24. Saat ini autentikasi yang dapat digunakan hanya password tipe 2, yaitu password sederhana dalam bentuk teks biasa.

Gambar 5.21 Format autentikasi RIPv2

Paket RIPv1 menggunakan paket broadcast yang membebani seluruh host dalam jaringan lokal. Untuk mengurangi beban host-host, paket RIPv2 dikirim menggunakan paket multicast dengan IP address 224.0.0.9. Selain itu paket RIPv2 juga dapat dikirim menggunakan paket broadcast untuk kompatibilitas dengan versi 1.

Dengan penambahan kemampuan pada RIPv2 di atas, secara ringkas perbedaan antara RIPv1 dengan RIPv2 dapat dilihat pada Tabel 5.9

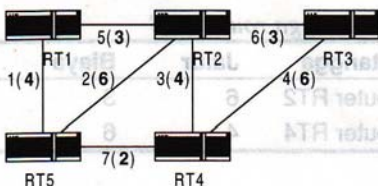
Tabel 5.9 Perbedaan RIPv2 dengan RIPv1

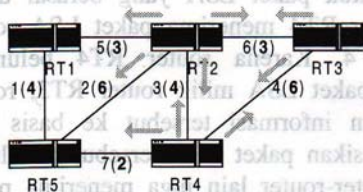
RIPv1	RIPv2
Tidak mendukung VLSM	Mendukung VLSM
Tidak aman, tanpa autentikasi	Menggunakan autentikasi
Rute dianggap internal jaringan	Membedakan rute internal dengan eksternal
Router pengirim paket selalu sebagai hop berikut	Dapat menggunakan hop berikut yang lain
Mem-broadcast paket	Menggunakan alamat multicast atau broadcast

5.9. Routing Link-State

Prinsip dasar routing *link-state* cukup sederhana. Setiap router mempunyai peta jaringan dan router kemudian menentukan rute ke setiap tujuan di jaringan berdasarkan peta tersebut. Peta jaringan disimpan router dalam bentuk basis data sebagai hasil dari pertukaran informasi link-state antara router-router bertetangga di jaringan tersebut. Setiap record dalam basis data menunjukkan status sebuah jalur dalam jaringan (*link-state*).

Routing link-state membentuk peta jaringan dalam tiga tahap. Tahap pertama, setiap router mengenali seluruh tetangganya. Tahap berikutnya, router-router saling bertukar informasi link-state, dan tahap terakhir setiap router menghitung jarak terpendek ke setiap tujuan.

Untuk menjelaskan protokol routing *link-state,* perhatikan jaringan seperti pada Gambar 5.10 yang diperlihatkan kembali pada Gambar 5.22. Pada gambar tersebut terdapat penambahan, yaitu setiap jalur diberi metrik yang menunjukkan biaya untuk melalui jalur tersebut (bilangan yang terletak di dalam tanda kurung). Sebuah jalur semakin disukai jika biayanya semakin kecil.

Gambar 5.22 Jaringan TCP/IP dengan biaya untuk tiap jalur

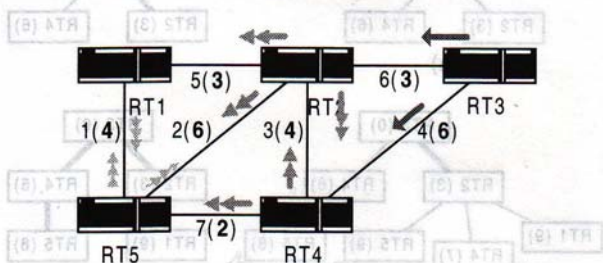
Pertama-tama yang dilakukan oleh router-router adalah saling memperkenalkan diri kepada tetangga masing-masing, yaitu dengan mengirimkan paket *hello* ke semua jalur yang terhubung dengannya. Kita perhatikan router RT3. Paket *hello* yang dikirimkan router RT3 kira-kira berbunyi seperti "*Hello*, saya RT3." Pada Gambar 5.23 terlihat router RT3 mengirimkan paket *hello* ke jalur 4 dan 6; RT2 mengirim ke jalur 2, 3, 5 dan 6; serta RT4 mengirim ke jalur 4 dan 7. Router RT3 kemudian mengetahui tetangga-tetangganya berdasarkan paket *hello* yang ia terima. Selain itu router RT3 juga mengetahui berapa biaya untuk mencapai setiap tetangga tersebut dan data-data ini disimpan dalam basis data seperti pada Tabel 5.10.

Gambar 5.23 Router RT3 mengirim dan menerima paket Hello

Tabel 5.10 Tetangga router RT3

Tetangga	Jalur	Biaya
Router RT2	6	3
Router RT4	4	6

Setelah router-router memiliki basis data mengenai tetangga masing-masing, routing link-state memasuki tahap kedua. Pada tahap ini router-router mengirim basis data tersebut kepada tetangga masing-masing dalam paket *link-state advertisement* (LSA). Router yang menerima paket LSA harus meneruskan paket tersebut ke seluruh router tetangganya yang lain dan juga memasukkan informasi dari paket LSA ke dalam basis data miliknya jika informasi tersebut lebih baru daripada yang terdapat dalam basis data. Jika paket LSA yang diterima router ternyata bukan paket yang baru, paket tersebut dibuang. Proses seperti ini berlangsung terus sampai setiap router di jaringan menerima paket LSA dari router yang lain. Proses di atas disebut sebagai *flooding* karena seolah-olah membanjiri jaringan dengan paket LSA.

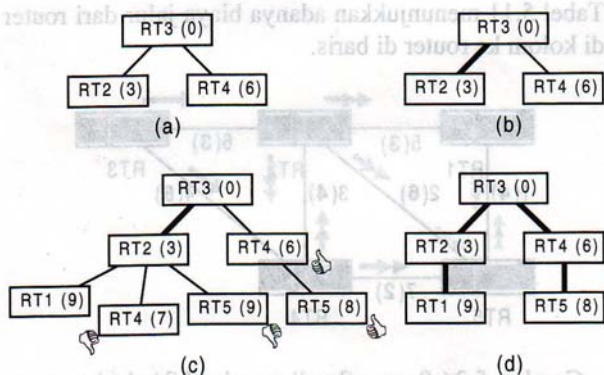
Gambar 5.24 memperlihatkan bagaimana proses *flooding* untuk paket LSA yang berasal dari router RT3. Router RT4 menerima paket LSA router RT3 dari jalur 4. Karena router RT4 belum pernah menerima paket LSA milik router RT3, router RT4 memasukkan informasi tersebut ke basis data dan mendistribusikan paket LSA tersebut melalui jalur 3 dan 7. Router-router lain juga menerima paket LSA milik router RT3 untuk pertama kali dan mereka melakukan hal yang sama seperti router RT4. Beberapa saat kemudian router RT2 menerima paket LSA milik

router RT3 dari router RT4 (router RT2 adalah tetangga router RT4). Router RT2 memperhatikan bahwa paket tersebut pernah diterimanya sehingga paket tersebut dibuang saja dan tidak perlu didistribusikan lagi. Dengan cara ini seluruh router tidak akan mengirimkan paket LSA yang sama dua kali. Hasil dari proses *flooding* adalah sebuah basis data *link-state* untuk jaringan yang sama di setiap router. Basis data *link-state* untuk jaringan Gambar 5.22 dapat dilihat pada Tabel 5.11. Setiap entri di Tabel 5.11 menunjukkan adanya biaya jalur dari router di kolom ke router di baris.

Gambar 5.24 Proses flooding paket LSA dari router RT3

Tabel 5.11 Basis data link-state untuk jaringan pada Gambar 5.22

ke \ dari	RT1	RT2	RT3	RT4	RT5
RT1		3			4
RT2	3		3	4	6
RT3		3		6	
RT4		4	6		2
RT5	4	6		2	

5.9.1. Menghitung rute terbaik

Setelah router mempunyai peta jaringan, router menghitung rute terbaik ke setiap tujuan di jaringan menggunakan algoritma Djikstra yang umum disebut sebagai *shortest path first* (SPF). Algoritma Djikstra membuat pohon dari jaringan dengan sistem yang melakukan perhitungan menjadi akar dari pohon tersebut. Contoh penggunaan algoritma SPF di router RT3 diperlihatkan menggunakan Gambar 5.25.

Gambar 5.25 Router RT3 melakukan algoritma SPF

Menghitung rute terpendek dari router RT3 dimulai dengan membuat pohon dengan router RT3 sebagai akar pohon dengan router RT2 dan RT4 menjadi daun (*leaves*) dari pohon tersebut (Gambar 5.25(a)). Biaya rute ke RT2 adalah yang paling kecil di antara route-rute lain dan rute tersebut dipastikan menjadi rute terpendek untuk ke RT2. Di Gambar 5.25 rute yang Rute lainnya (rute ke RT4) dianggap sebagai rute terpendek tentatif. Kemudian pohon jaringan diperluas

dengan memasukkan tetangga router RT2, yaitu RT1, RT4 dan RT5. Hasil perbandingan menunjukkan rute RT4(7) harus dihapus karena biayanya lebih panjang daripada RT4(6) (diberi tanda ♘)sehingga dihapus dari pohon. Rute ke RT1 dan RT5 melalui RT2 dianggap sebagai rute terpendek sementara.

Pohon diperluas lagi dengan menambahkan tetangga dari router RT1, tetapi RT1 tidak memiliki tetangga yang lain lagi. Pohon diperluas dengan tetangga dari router RT5, tetapi tidak ada tetangga lagi. Perluasan pohon sekarang beralih pada router RT4 yaitu dengan menambahkan router RT5 dan menjadikannya rute terpendek sementara. Hasil perbandingan rute menuju RT5 menyebabkan rute RT5(9) (diberi tanda ♘) dihapus dari pohon. Karena pohon tidak dapat diperluas lagi, maka seluruh rute terpendek sementara dipastikan menjadi rute terpendek untuk pohon tersebut (Gambar 5.25.(e)). Dengan demikian algoritma SPF untuk router RT3 selesai sampai di sini.

5.9.2. Perubahan kondisi jaringan

Routing link-state juga saling bertukar informasi dalam selang tertentu untuk mengetahui kondisi terakhir jaringan. Informasi tersebut adalah dalam bentuk paket *hello* yang berguna untuk memberitahu bahwa router masih aktif. Sebuah router akan menganggap router tetangganya mati jika tidak lagi mendengar paket *hello* dari router tersebut setelah selang waktu tertentu. Misalkan interval pertukaran paket *hello* antara router RT1 dan RT2 adalah 30 detik dan jika router RT2 tidak mendengar paket *hello* RT1 setelah 120 detik, maka jalur ke router RT1 dianggap putus.

Perubahan jaringan menyebabkan basis data *link-state* berubah. Basis data yang pertama kali berubah adalah basis data pada router yang berdekatan dengan jalur yang berubah tersebut. Router harus menginformasikan perubahan ke router-router lain menggunakan paket LSA menggunakan *flooding*. Akibat perubahan ini tentu saja router *link-state* harus kembali menghitung jarak terpendek ke setiap tujuan di jaringan.

Konvergensi jaringan akibat perubahan membutuhkan waktu yang ditentukan oleh kecepatan proses *flooding* dan penghitungan SPF. Kecepatan penghitungan SPF tergantung seberapa besar jaringan yang digunakan dan kecepatan proses *flooding* tergantung pada delay di jaringan. Router-router *link-state* umumnya menjalankan tahap *flooding* dan penghitungan SPF secara bersamaan untuk mempercepat konvergensi. Dengan demikian routing *link-state* dapat dengan cepat merespon perubahan kondisi jaringan.

5.10. Open Shortest Path First versi 2

Protokol routing Open Shortest Path First (OSPF) dirancang dan dikembangkan untuk jaringan TCP/IP oleh sebuah kelompok kerja OSPF. Spesifikasi terakhir OSPF versi 2 yang dihasilkan kelompok kerja OSPF tertulis dalam dokumen RFC 2178. OSPF mendukung jaringan *point-to-point*, *point-to-multipoint*, dan jaringan multiakses. Protokol ini termasuk kategori routing *link-state*, oleh sebab itu OSPFv2 (selanjutnya disebut sebagai OSPF saja) dapat menghitung jalur routing yang pasti terbebas dari *loop*. Beberapa kelebihan OSPF lainnya antara lain dapat dengan cepat mendeteksi perubahan yang terjadi di jaringan dan

menjadikan routing kembali konvergen dalam waktu yang singkat dengan sedikit pertukaran data. OSPF mampu menangani routing jaringan TCP/IP yang besar dan membuat hierarki routing dengan membagi jaringan menjadi beberapa area. Paket-paket OSPF dikirim dan diterima dengan memanfaatkan IP muticast. Setiap paket routing OSPF menggunakan autentikasi untuk meningkatkan keamanan. Kelemahan protokol OSPF adalah membutuhkan kemampuan CPU dan memori yang relatif besar.

OSPF adalah protokol routing yang kompleks, terbukti dari dokumen spesifikasi yang tebalnya lebih dari 200 halaman. Kita akan membicarakan hal-hal penting dalam protokol ini dan berusaha menghindari detailnya yang rumit. Jika ingin mengetahui detail protokol ini, pembaca dipersilahkan untuk membuka dokumen RFC untuk OSPFv2 (saat buku ini dibuat dokumen tersebut adalah RFC 2178).

Proses dasar dalam routing OSPF adalah menghidupkan *adjacency*, proses *flooding*, dan penghitungan tabel routing. Router-router mengirimkan paket *Hello* ke seluruh jaringan yang terhubung dengannya secara periodik. Jika paket *Hello* sebuah router tidak terdengar setelah selang waktu tertentu, router tersebut dianggap mati. Selang waktu ini secara default ditentukan empat kali interval pengiriman paket *Hello*.

Router-router selalu berusaha *adjacent* dengan router tetangganya berdasarkan paket *Hello* yang diterima. Dalam jaringan multi akses, router-router memilih *Designated Router* dan *Backup Designated Router* dan mencoba *adjacent* dengan kedua router tersebut. Jika *Designated Router* atau *Backup Designated Router*

mati, router-router kembali melakukan pemilihan untuk menggantikan router yang mati tersebut. Basis data jaringan multi akses menjadi lebih sederhana dengan menggunakan *Designated Router* dibandingkan jika setiap router dalam jaringan tersebut harus adjacent dengan setiap router lain.

Proses *flooding* dimulai ketika router memiliki router tetangga yang *adjacent* dan proses ini hanya terjadi antara router-router yang *adjacent* dengan saling bertukar LSA-LSA yang terbaru saja. Dengan cara ini proses *flooding* tidak memberatkan jaringan dengan paket LSA. Proses *flooding* terjadi ketika dalam jaringan terdapat LSA yang baru. Router-router mengirimkan LSA baru mungkin karena *link-state*-nya berubah (misalnya sebuah jalur terputus) atau karena LSA miliknya sudah kadaluarsa. Sebuah LSA menjadi kadaluwarsa dalam setengah jam, jadi setidaknya setiap setengah jam terjadi *flooding* di jaringan.

Setiap kali basis data link-state router berubah, router kembali perlu menghitung rute terbaik dan membentuk tabel routing baru. Yang dimaksud dengan rute terbaik adalah rute dengan biaya terendah pada *shortest path tree* (pohon jalur terpendek) jaringan. Spesifikasi OSPF tidak mensyaratkan bahwa router harus menjalankan algoritma Djikstra untuk memperoleh pohon tersebut sepanjang pohon yang dihasilkan sama dengan hasil algoritma Djikstra.

5.10.1. Basis Data Link-State

OSPF sebagai routing link-state menyimpan peta jaringan dalam bentuk basis data link-state. OSPF mengenal tiga jenis jaringan dalam membuat basis data

link-state: jaringan point-to-point, jaringan stub, dan jaringan broadcast atau NBMA. Basis data link-state dinyatakan dalam bentuk sebuah directed graph dengan router dan jaringan digambarkan sebagai verteks. Graph membuat edge dari verteks A ke RT2 jika dan hanya jika A terhubung dengan RT2 (digambar diberi tanda +). Gambar 5.26 menunjukkan representasi jaringan dasar oleh basis data link-state OSPF. Dalam sebuah jaringan yang besar, basis data link-state OSPF merupakan gabungan dari basis data kecil seperti Gambar 5.26.

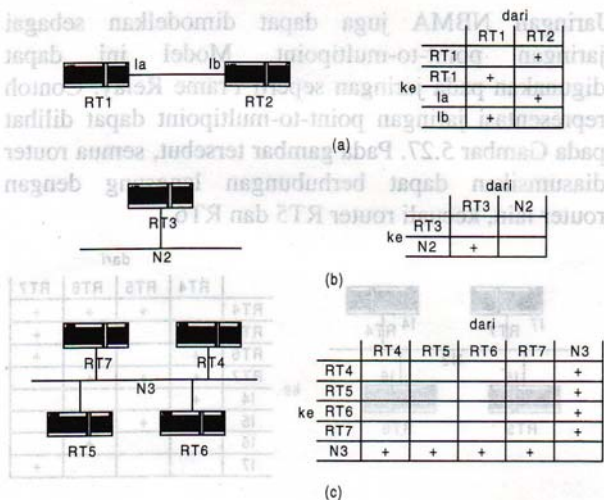
(a)

ke \ dari	RT1	RT2
RT1		+
RT1	+	
Ia		+
Ib	+	

(b)

ke \ dari	RT3	N2
RT3		
N2	+	

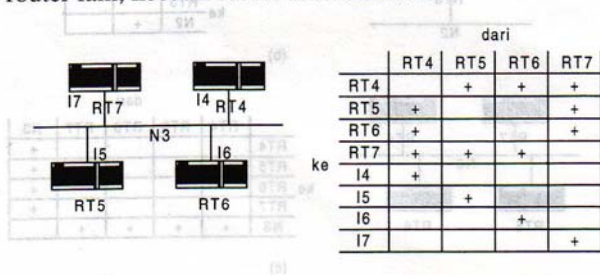
(c)

ke \ dari	RT4	RT5	RT6	RT7	N3
RT4					+
RT5					+
RT6					+
RT7					+
N3	+	+	+	+	

Gambar 5.26 Peta jaringan dasar OSPF

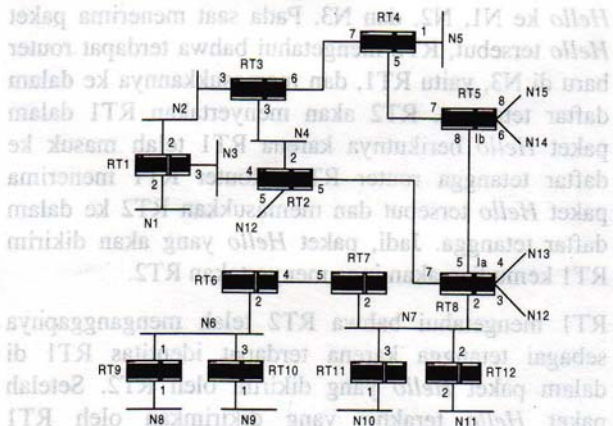
Pada Gambar 5.26(a) terlihat dua buah router RT1 dan RT2 yang terhubung oleh jaringan point-to-point masing-masing melalui interface Ia dan Ib. Kedua

interface tidak perlu diberi IP address, tetapi jika interface diberi IP address, maka representasinya berubah menjadi dua buah jaringan stub. Gambar 5.26(b) memperlihatkan jaringan stub, yaitu jaringan yang hanya memiliki sebuah router saja. Datagram IP yang melalui router RT3 pasti berawal atau berakhir di jaringan N3. Gambar 5.26(c) memperlihatkan jaringan broadcast atau *non-broadcast multiple access* (NBMA) dengan beberapa router. Jaringan seperti ini dapat menjadi jaringan transit bagi datagram IP yang berasal dan menuju jaringan lain.

Jaringan NBMA juga dapat dimodelkan sebagai jaringan point-to-multipoint. Model ini dapat digunakan pada jaringan seperti Frame Relay. Contoh representasi jaringan point-to-multipoint dapat dilihat pada Gambar 5.27. Pada gambar tersebut, semua router diasumsikan dapat berhubungan langsung dengan router lain, kecuali router RT5 dan RT6.

		dari			
		RT4	RT5	RT6	RT7
ke	RT4		+	+	+
	RT5	+			+
	RT6	+			+
	RT7	+	+	+	
	I4	+			
	I5		+		
	I6			+	
	I7				+

Gambar 5.27 NBMA sebagai jaringan point-to-multipoint

Perhatikan Gambar 5.28 yang menunjukkan sebuah jaringan yang cukup besar. Gambar 5.28 kita jadikan contoh untuk melihat bagaimana protokol OSPF

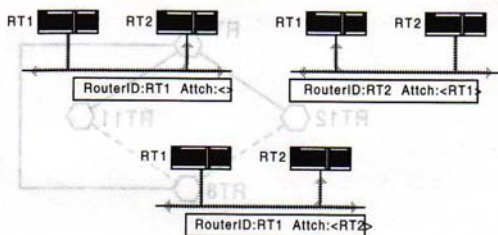
bekerja. Pada gambar tersebut diperlihatkan angka-angka yang menunjukkan biaya untuk melewatkan data keluar melalui interface router tersebut. Jaringan N12 – N15 yang terhubung dengan router RT5 dan RT8 digambarkan tidak dengan detail karena jaringan tersebut tidak menggunakan protokol routing OSPF. Jaringan tersebut dapat diperoleh dari protokol routing lain, misal RIP, BGP atau statik.

Gambar 5.28 Contoh jaringan TCP/IP

5.10.2. Menghidupkan Adjacency

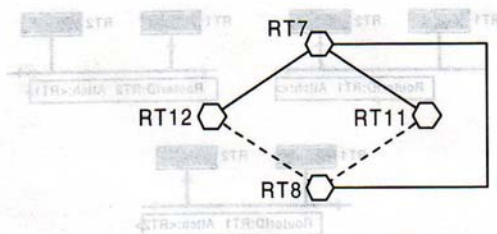
Pada saat sebuah router baru menjalankan protokol OSPF, router tersebut tidak mengetahui apa pun mengenai tetangga-tetangganya. Router kemudian mulai mengirimkan paket *Hello* ke seluruh interface jaringan untuk memperkenalkan dirinya. Paket *Hello* dikirimkan secara teratur ke jaringan untuk mengetahui

router-router apa saja yang aktif. Router juga mengirimkan identitas router-router tetangga yang dapat didengarnya di dalam paket *Hello*. Bila sebuah router menerima paket *Hello* yang mengandung identitas dirinya, maka router tersebut mengetahui bahwa ia dapat melakukan hubungan dua arah dengan router pengirim paket *Hello*.

Misal router RT1 baru menyala dan mengirim paket *Hello* ke N1, N2, dan N3. Pada saat menerima paket *Hello* tersebut, RT2 mengetahui bahwa terdapat router baru di N3, yaitu RT1, dan memasukkannya ke dalam daftar tetangga. RT2 akan menyertakan RT1 dalam paket *Hello* berikutnya karena RT1 telah masuk ke daftar tetangga router RT2. Router RT1 menerima paket *Hello* tersebut dan memasukkan RT2 ke dalam daftar tetangga. Jadi, paket *Hello* yang akan dikirim RT1 kemudian akan juga menyertakan RT2.

RT1 mengetahui bahwa RT2 telah menganggapnya sebagai tetangga karena terdapat identitas RT1 di dalam paket *Hello* yang dikirim oleh RT2. Setelah paket *Hello* terakhir yang dikirimkan oleh RT1 diterima, RT2 mengetahui bahwa RT1 juga telah menganggapnya sebagai tetangga. Jika RT2 telah memasukkan RT1 ke dalam daftar tetangga, dan RT1 juga memuat RT2 di dalam daftar tetangga, maka kedua router mengetahui bahwa mereka dapat melakukan komunikasi dua arah.

Gambar 5.29 RT1 dan RT2 mengirim paket Hello *ke jaringan N3*

Router akan menghidupkan adjacency dengan router tetangganya setelah komunikasi dua arah terjadi. Router-router yang terhubung oleh jaringan point-to-point, point-to-multipoint, dan *virtual link* selalu adjacent. Pada jaringan broadcast dan NBMA router-router harus memilih sebuah Designated Router (DR) dan sebuah Backup Designated Router (BDR) dari semua router di dalam jaringan tersebut. Router-router di jaringan memilih DR dan BDR berdasarkan bilangan prioritas router yang terdapat dalam paket *Hello* router. Router-router lain dalam jaringan kemudian menghidupkan adjacency dengan DR dan BDR terpilih.

Sebagai contoh, router RT7 dan RT8 masing-masing terpilih menjadi DR dan BDR untuk jaringan N7. Dengan kondisi seperti ini, RT11 dan RT12 yang juga terdapat di N7 akan adjacent dengan RT7 dan RT8. Gambar 5.30 memperlihatkan graph adjacency untuk router-router di jaringan N7.

Gambar 5.30 Graph adjacency di jaringan N7

5.10.3. Sinkronisasi basis data

Spesifikasi OSPF menyatakan bahwa hanya router yang adjacent yang harus tetap sinkron satu sama lain. Proses sinkronisasi dimulai sejak router mencoba menghidupkan adjacency yaitu saat router dapat berkomunikasi dua arah dan telah terpilih sebuah Designated Router. Dua router yang mencoba menjadi adjacent saling mengirim paket *Database Description* yang memberitahukan LSA router masing-masing. Fungsi paket *Database Description* adalah untuk mengetahui LSA-LSA terbaru di antara kedua router. Setelah itu kedua router mulai melakukan pertukaran basis data sehingga setiap router memiliki LSA yang terbaru. Paket-paket LSA yang dipertukarkan tersebut dikirim ke IP address router atau menggunakan IP address multicast pada jaringan broadcast. Dua router kemudian menjadi adjacent sepenuhnya ketika kedua router itu telah memiliki semua LSA terbaru.

OSPF memulai proses *flooding* segera setelah proses pertukaran basis data dimulai. Segera setelah LSA terbaru diterima oleh sebuah router, LSA tersebut langsung dikirimkan ke router lain yang adjacent.

Perhatikan RT3 di Gambar 5.28 yang akan adjacent dengan RT1, RT2, dan RT4. Ketika menerima LSA terbaru dari RT4, RT3 kemudian segera mengirimkan LSA tersebut ke RT1 dan RT2. Demikian pula jika menerima LSA terbaru dari RT1, RT3 segera mengirimkan LSA tersebut ke RT2 dan RT4. Dengan cara ini sinkronisasi basis data di seluruh jaringan menjadi lebih mudah dan lebih cepat selesai.

5.10.4. Link State Advertisement

Pada saat sinkronisasi basis data dan prosedur flooding, router-router yang adjacent saling bertukar *link-state advertisement* (LSA). OSPF mengenal lima tipe LSA. LSA tipe 1 adalah router-LSA; LSA ini berasal dari seluruh router dan menunjukkan keadaan (state) interface router yang menuju sebuah area. Flooding LSA ini terbatas hanya dalam sebuah area saja. Jika jaringan tidak dibagi menjadi backbone dan area, maka LSA tipe 1 disebarkan ke seluruh jaringan. Tabel 5.12 menunjukkan Router-LSA yang berasal dari router RT3

Tabel 5.12 Router-LSA dari RT3

		dari			
		RT3	RT4	N2	N4
ke	RT3				
	RT4	6			
	N2	3			
	N4	1			

LSA tipe 2, network-LSA, berasal dari jaringan broadcast dan NBMA. LSA ini dikirimkan oleh Designated Router jaringan bersangkutan dan berisi

daftar router yang terhubung ke jaringan tersebut. LSA tipe ini juga hanya disebarkan dalam satu area saja. Contoh network-LSA dapat dilihat pada Tabel 5.13.

Tabel 5.13 Network-LSA jaringan N6

		dari			
		RT6	RT9	RT10	N6
ke	RT6				0
	RT9				0
	RT10				0
	N6				

Summary-LSA, LSA tipe 3 dan 4, berasal dari Area Border Router (ABR) dan disebarkan ke area LSA yang bersangkutan. Setiap summary-LSA berisi rute ke sebuah tujuan di luar area. LSA tipe 3 berisi rute ke jaringan dan LSA tipe 4 berisi rute ke ke AS Boundary Router (ASBR). Konsep area, ABR, dan summary-LSA akan kita bicarakan pada pembagian jaringan menjadi area.

LSA tipe 5 adalah AS-external-LSA yang berasal dari ASBR dan disebarkan ke seluruh autonomous system. Setiap LSA ini berisi rute ke sebuah tujuan di luar autonomous system. Pada setiap LSA ini juga terdapat alamat *forwarding* sehingga rute eksternal dapat dibelokkan melalui router yang disebut pada alamat *forwarding*, dan bukan melalui router asal LSA.

Kelima tipe LSA di atas menjadi komponen dasar dalam membentuk basis data link state sebuah autonomous system. Tabel 5.14 berikut memperlihatkan basis data link state jaringan pada Gambar 5.26.

Tabel 5.14 Basis data link-state jaringan Gambar 5.26

ke \ dari	RT1	RT2	RT3	RT4	RT5	RT6	RT7	RT8	RT9	RT10	RT11	RT12	N2	N3	N4	N6	N7
RT1													0	0			
RT2								7						0	0		
RT3													0		0		
RT4			6		7												
RT5				5				5									
RT6				7			5									0	
RT7						4											0
RT8		5			8												0
RT9																0	
RT10																0	
RT11																	0
RT12																	0
N1	2																
N2	2		3														
N3	3	4															
N4		2	3														
N5				1													
N6						2			1	3							
N7							2	2			3	2					
N8									1								
N9										1							
N10											1						
N11												2					
N12			5					3									
N13								4									
N14					6												
N15					8												

5.10.5. Penghitungan Tabel Routing

Setelah memiliki basis data seperti di atas, setiap router menghitung pohon jalur terpendek (*shortest path tree*, SPT) dengan dirinya sebagai akar pohon tersebut. Contoh SPT untuk salah satu router di jaringan diperlihatkan pada Gambar 5.31. Router kemudian membentuk tabel routing berdasarkan pohon tersebut. Sebagian tabel routing pada router RT4 adalah seperti terlihat pada Tabel 5.15.

Proses pembentukan tabel routing OSPF secara singkat dapat dibagi menjadi beberapa langkah:

1. Menghapus tabel routing. Setiap kali menghitung tabel routing, router menyimpan tabel routing yang

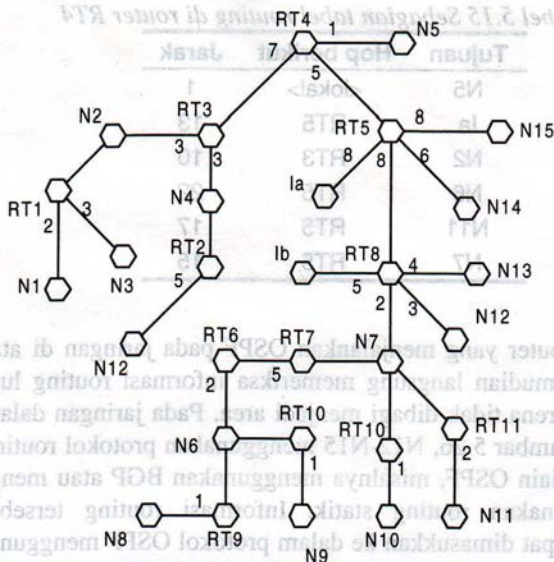
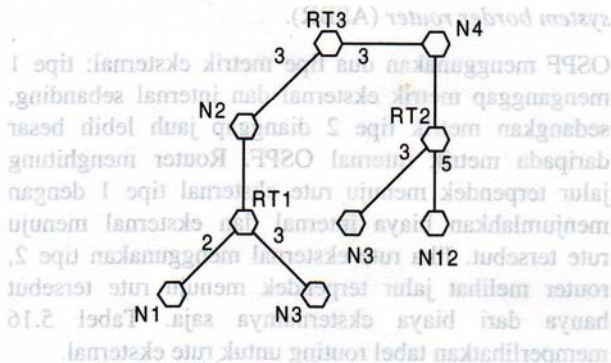
lama dan membangun tabel routing yang baru dari nol. Tabel routing yang lama disimpan untuk mengetahui perubahan tabel routing.

2. Menghitung rute intraarea dengan membuat pohon jalur terpendek untuk setiap area yang terhubung dengan router itu.

3. Menghitung rute inter-area dengan memeriksa summary-LSA. *Area Border Router* hanya memeriksa summary-LSA untuk backbone.

   Area Border Router yang terhubung dengan area transit memeriksa summary-LSA untuk mencari rute terbaik melewati area transit tersebut.

4. Menghitung rute eksternal dengan memeriksa AS-external-LSA.

Penghitungan SPT untuk jaringan di atas tidak menghasilkan sebuah tujuan dengan dua jalur yang memiliki jumlah biasa yang sama. Apabila terdapat sebuah jaringan yang dapat dicapai melalui dua atau lebih jalur yang biayanya sama, OSPF akan menggunakan seluruh jalur tersebut untuk mencapai tujuan. Kasus seperti ini disebut sebagai *equal cost mutipath*. Misal biaya dari router RT2 ke N3 diganti menjadi 3, bukan 4. Dengan biaya ini, dari RT3 akan terdapat dua jalur menuju N3 dengan biaya yang sama, yaitu 6. Pohon SPF untuk RT3 yang menuju ke N3 akan menjadi seperti pada Gambar 5.32.

Gambar 5.31 Pohon jalur terpendek untuk router RT4

Gambar 5.32 Equal cost multipath dari RT3 ke N3

Tabel 5.15 Sebagian tabel routing di router RT4

Tujuan	Hop berikut	Jarak
N5	<lokal>	1
Ia	RT5	13
N2	RT3	10
N6	RT5	22
N11	RT5	17
N7	RT5	15

Router yang menjalankan OSPF pada jaringan di atas kemudian langsung memeriksa informasi routing luar karena tidak dibagi menjadi area. Pada jaringan dalam Gambar 5.26, N12-N15 menggunakan protokol routing selain OSPF, misalnya menggunakan BGP atau menggunakan routing statik. Informasi routing tersebut dapat dimasukkan ke dalam protokol OSPF menggunakan LSA tipe 5, AS-external-LSA. OSPF menganggap semua jaringan yang tidak menggunakan protokol OSPF berada di luar autonomous system dan router tempat LSA tipe 5 berasal disebut sebagai *autonomous system border router* (ASBR).

OSPF menggunakan dua tipe metrik eksternal: tipe 1 menganggap metrik eksternal dan internal sebanding, sedangkan metrik tipe 2 dianggap jauh lebih besar daripada metrik internal OSPF. Router menghitung jalur terpendek menuju rute eksternal tipe 1 dengan menjumlahkan biaya internal dan eksternal menuju rute tersebut. Jika rute eksternal menggunakan tipe 2, router melihat jalur terpendek menuju rute tersebut hanya dari biaya eksternalnya saja. Tabel 5.16 memperlihatkan tabel routing untuk rute eksternal.

Tabel 5.16 Tabel routing RT4 menuju jalur eksternal

Tujuan	Hop berikut	Jarak	Tujuan	Hop Berikut	Jarak
N12	RT3	15	N12	RT5	3
N13	RT5	17	N13	RT5	4
N14	RT5	11	N14	RT5	6
N15	RT5	13	N15	RT5	8

a. LSA tipe 1 *b. LSA tipe 2*

Di Gambar 5.26 terlihat bahwa terdapat dua jalur yang dapat digunakan menuju rute eksternal N12, yaitu melalui RT2 dan RT8. Jika AS-external-LSA N2 diberi tipe 1 oleh kedua router, jalur terpendek menuju N12 dari RT4 adalah 15 yaitu melalui RT2. Jika router-router menggunakan AS-external-LSA tipe 2 untuk N12, maka jalur terpendek dari RT4 adalah melalui RT8, yaitu 3.

Kedua tipe LSA eksternal dapat dimasukkan bersama-sama ke dalam jaringan. Apabila ada dua rute eksternal yang sama tetapi menggunakan tipe LSA yang berbeda, maka OSPF selalu mendahulukan metrik LSA eksternal tipe 1.

5.10.6. OSPF dengan Area dan Backbone

Protokol routing OSPF dapat membentuk hierarki routing, yaitu dengan membagi jaringan dalam beberapa area. Setiap area menjalankan algoritma link-state yang hanya meliputi area tersebut. Setiap router dalam sebuah area mengetahui topologi jaringan untuk area tersebut dan tidak mengetahui topologi jaringan area lain. Batas area adalah router-router yang terhubung ke beberapa area dan disebut sebagai router perbatasan area (*area border router*, ABR).

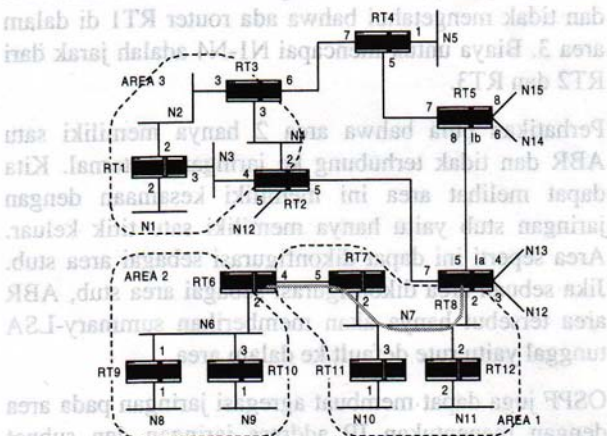
Backbone adalah area khusus yang harus selalu ada dalam jaringan. Backbone diberi nomor identitas area 0. Router-router yang terletak dalam area ini disebut sebagai router backbone. Backbone berfungsi untuk mendistribusikan informasi routing ke seluruh area dan seluruh ABR harus terhubung dengan backbone. Backbone harus selalu terhubung, walaupun tidak harus secara fisik. Backbone dapat terhubung menggunakan jalur virtual (*virtual link*) antara dua router backbone dengan interface yang terhubung ke area yang sama. OSPF menganggap jalur virtual ini sebagai jalur point-to-point yang tidak diberi IP address. Biaya untuk melewati jalur virtual diperoleh dari jarak intra-area kedua router.

Membagi jaringan menjadi beberapa area berarti memperkecil jumlah informasi routing dan mempermudah router melakukan penghitungan jalur terpendek karena basis data link-state menjadi lebih kecil. Dengan adanya area maka sekarang terdapat dua tingkat routing: routing intra-area untuk datagram yang berasal dari dan menuju ke area yang sama; dan routing inter-area untuk datagram yang melintasi area yang berbeda.

Gambar 5.33 memperlihatkan jaringan pada Gambar 5.26 yang dibagi menjadi backbone dengan tiga area. Area 1 terdiri atas jaringan N7, N10, N11 dan jalur point-to-point antara router RT7 dan RT6; beserta router-router yang menghubungkannya. Area 2 terdiri atas N6, N8, dan N9 dengan router-routernya. Area 3 terdiri atas N1-N4 beserta router-routernya.

Router-router RT1, RT7, RT9, RT10, RT11, dan RT12 adalah router internal yaitu router-router yang terletak

dalam satu area saja. Router RT2, RT3, RT6, dan RT8 adalah router perbatasan area. Router RT2, RT5, dan RT8 adalah router perbatasan AS (AS boundary router, ASBR).

Pada Gambar 5.33 terlihat bahwa area 2 berbatasan dengan area 1 dan tidak berhubungan langsung secara fisik dengan backbone. Router RT6 adalah router perbatasan area dan menurut spesifikasi OSPF setiap ABR harus terhubung dengan backbone. Untuk memenuhi spesifikasi tersebut, dibuat jalur virtual antara RT6 dengan RT8 dan area 1 bertindak sebagai area transit. Dalam gambar, jalur virtual ini ditunjukkan dengan jalur yang berwarna abu-abu. Dengan jalur virtual antara RT6 dan RT8 backbone menjadi terhubung dan semua ABR terhubung dengan backbone.

Gambar 5.33 Jaringan dengan tiga area OSPF

Pada jaringan yang tidak dibagi menjadi beberapa area, basis data link-state sama di semua router. Pada jaringan seperti Gambar 5.33 terdapat perbedaan basis data link-state di router-router. Basis data ini hanya sama pada router yang terletak di area yang sama. Router perbatasan area memiliki beberapa basis data link-state, tergantung jumlah area yang terhubung dengan router tersebut.

Pada jaringan ini sekarang terdapat paket summary-LSA yang berasal dari setiap ABR. Paket summary-LSA menyembunyikan topologi area. Paket summary-LSA yang dikirimkan ABR ke backbone berisi jaringan-jaringan yang terdapat dalam area. Tabel 5.18 memperlihatkan summary-LSA yang dikirimkan dari area 3 ke backbone. Dari summary-LSA tersebut, router-router di luar area 3 mengetahui bahwa untuk mencapai jaringan N1-N4 dapat melalui RT2 atau RT3 dan tidak mengetahui bahwa ada router RT1 di dalam area 3. Biaya untuk mencapai N1-N4 adalah jarak dari RT2 dan RT3.

Perhatikan pula bahwa area 2 hanya memiliki satu ABR dan tidak terhubung ke jaringan eksternal. Kita dapat melihat area ini memiliki kesamaan dengan jaringan stub yaitu hanya memiliki satu titik keluar. Area seperti ini dapat dikonfigurasi sebagai area stub. Jika sebuah area dikonfigurasi sebagai area stub, ABR area tersebut hanya akan memberikan summary-LSA tunggal yaitu rute default ke dalam area.

OSPF juga dapat membuat agregasi jaringan pada area dengan menentukan IP address jaringan dan subnet mask yang meliputi semua jaringan di area tersebut. Jika pada konfigurasi ABR diberikan jangkauan

(*range*) IP address untuk area maka summary-LSA memuat satu jaringan saja, yaitu IP address dan subnet mask untuk jaringan tersebut. Biaya yang disertakan di summary-LSA adalah biaya maksimum jaringan. Misal area 1 dikonfigurasi untuk membuat summary-LSA hanya terdiri atas satu LSA saja, maka summary-LSA dikeluarkan oleh RT6 untuk jaringan N6, N8, N9 dengan biaya 3.

Tabel 5.17 Summary-LSA jaringan area 3 yang dikirim ke backbone

Jaringan	LSA RT2	LSA RT3
N1	6	5
N2	6	3
N3	4	7
N4	2	3

Setiap router perbatasan area (ABR) di jaringan mendengar summary-LSA dari ABR yang lain melalui backbone. Tabel 5.18 memperlihatkan basis data link-state untuk backbone. ABR setiap area kemudian menghitung jarak dari ABR ke setiap jaringan di area lain dengan menjumlahkan jarak dalam summary-LSA dengan jarak backbone. Jaringan-jaringan tersebut kemudian dimasukkan ke dalam summary-LSA untuk area. Di samping itu, ABR juga memasukkan ASBR ke dalam summary-LSA.

Setelah router-router di area menerima LSA dan membentuk basis data link-state untuk area tersebut, router akan menghitung jalur terpendek ke tujuan seperti penghitungan jalur terpendek dalam jaringan tanpa area.

Tabel 5.18 Basis data link-state untuk backbone

	dari					
	RT2	RT3	RT4	RT5	RT6	RT8
RT2						7
RT3			7			
RT4		6		7		
RT5						
RT6						7
RT7					4	2
RT8	5		5		6	
N1	6	5				
N2	6	4				
N3	4	7				
N4	2	3				
ke N5			1			
N6,N8,N9					3	
N7					6	2
N10					7	3
N11					7	3
N12	5					3
N13						4
N14				6		
N15				8		
Ia				8		
Ib						5

5.11. Ringkasan

Routing adalah proses yang penting dalam penyampaian datagram di jaringan TCP/IP. Pengaturan routing dapat menentukan kinerja sebuah jaringan. Pembentukan tabel routing di router-router dalam jaringan dapat dilakukan secara manual atau secara otomatis melalui protokol routing. Pembentukan tabel routing secara manual (routing statik) cocok digunakan pada jaringan yang kecil sementara penggunaan protokol routing (routing dinamik) untuk jaringan yang besar.

Routing Information Protocol (RIP) sangat sederhana dan lambat mengetahui perubahan kondisi jaringan. Protokol ini terdapat dalam dua versi, RIPv1 dan RIPv2. RIPv2 menambah beberapa kemampuan RIPv1 seperti autentikasi, panjang subnet mask yang bervariasi, serta multicast. Walaupun demikian, RIPv2 tidak melakukan pengubahan mendasar pada algoritma vektor-jarak yang digunakan RIPv1.

Open Shortest Path First (OSPF) merupakan protokol routing yang kompleks. Setiap router yang menjalankan OSPF menyimpan peta jaringan dan menghitung jarak terpendek menuju semua tujuan di jaringan berdasarkan peta tersebut. Dengan cara ini router-router tidak akan menyebabkan *routing loop*. OSPF dapat membagi jaringan menjadi beberapa area. Pembagian ini sangat berguna pada jaringan yang besar karena OSPF membutuhkan kemampuan CPU dan memori yang besar. Saat ini OSPF direkomendasikan untuk mengganti RIP sebagai *interior gateway protocol* karena memiliki kemampuan yang jauh lebih baik daripada RIP.

Bab 6
Implementasi Routing di Jaringan

Pada bab lalu kita telah membicarakan konsep routing di jaringan TCP/IP, proses pembentukan tabel routing, dan protokol routing. Pada bab ini kita akan masuk ke implementasi konsep yang telah dibahas sebelumnya di jaringan TCP/IP menggunakan komputer dengan sistem operasi FreeBSD sebagai router. Bab ini dimulai dengan melihat sekilas perintah-perintah di FreeBSD yang berhubungan dengan Internet dan routing, serta membentuk routing jaringan secara manual. Kemudian kita akan menggunakan daemon routing GateD (dibaca gate-d) untuk menjalankan protokol routing RIP dan OSPF di jaringan.

Jaringan yang akan digunakan sebagai contoh implementasi routing adalah yang terlihat di Gambar 6.1. Gambar 6.1 merupakan jaringan pada Gambar 4.26 yang diberi alamat IP dan sedikit perubahan. Alamat IP dan jaringan yang tertulis di router dan jaringan adalah bagian host dari jaringan kelas B 132.92/16. Contoh: alamat IP *anubis* adalah 132.92.121.2, dalam Gambar 6.1 ditulis sebagai 121.2. Pengecualian bagi konvensi di atas adalah jaringan 171.21.0.0/16 yang berada di luar jaringan 132.92. Router-router di jaringan ini pun

sekarang telah diberi nama. Keterangan mengenai nama router-router tersebut dapat dilihat pada bab mengenai DNS.

Dalam gambar tersebut terlihat jaringan 132.92.0.0/16 yang terhubung dengan jaringan 171.21.0.0/16 dan Internet melalui router *nephthys*. Jaringan 132.92/16 terdiri atas dua tipe jaringan yaitu jaringan multiakses dan point-to-point. Semua interface jaringan point-to-point diberi alamat IP dan subnet mask untuk interface ini adalah 255.255.255.252, yaitu subnet mask terkecil yang mungkin. Jaringan 132.92.15.0/24, 132.92.63.0/24 dan 171.21.0.0/16 tidak digambarkan dengan detail karena rute ke jaringan-jaringan tersebut akan dimasukkan secara manual di seluruh bagian bab ini.

Gambar 6.1 Contoh Jaringan TCP/IP untuk implementasi routing

6.1. Perintah-Perintah FreeBSD untuk Internet

FreeBSD memiliki sejumlah perintah yang berkaitan dengan jaringan komputer TCP/IP. Kita akan melihat sepintas perintah-perintah yang berkaitan dengan interface jaringan dan routing. Secara umum perintah-perintah ini mirip dengan perintah sistem operasi Unix lainnya. Berikut ini adalah perintah-perintah tersebut yang disusun menurut abjad.

6.1.1. arp

Perintah ini untuk melihat dan mengubah isi tabel cache ARP.

Contoh penggunaan perintah:

Melihat entri ARP sebuah host

```
root:/
mehyt# arp 132.92.36.111
hetau.cnrg.net (132.92.36.111) at 0:80:ad:a7:a3:8a
```

Melihat seluruh isi tabel cache ARP

```
root:/
mehyt# arp -a
djed.cnrg.net (132.92.36.107) at 0:80:ad:17:96:34
mehyt.cnrg.net (132.92.36.108) at 0:20:4c:30:29:29
permanent
hetau.cnrg.net (132.92.36.111) at 0:80:ad:a7:a3:8a
kheper.cnrg.net (132.92.36.114) at 0:80:ad:a7:96:f5
```

Menghapus entri ARP sebuah host

```
root:/
mehyt# arp –d 132.92.36.114
132.92.36.114 (132.92.36.114) deleted
```

Mengganti isi entri ARP sebuah host

```
root:/
mehyt# arp –s 132.92.36.114 0:80:ad:a6:b6:65
```

6.1.2. netstat

Perintah ini untuk menunjukkan status jaringan.

Contoh penggunaan:

Melihat tabel routing

```
mehyt# netstat -nr
```

Tabel routing di FreeBSD terlihat seperti berikut

Routing tables

Internet:

Destination	Gateway	Flags	Refs	Use	Netif	Expire
default	132.92.36.125	UGSc	561	3799	ed0	
127.0.0.1	127.0.0.1	UH	0	6282	lo0	
132.92.21.121	127.0.0.1	UH	0	0	lo0	
132.92.21.122	132.92.21.121	UH	0	0	tun0	
132.92.122.36	132.92.36.122	UGHD	0	784	ed0	
132.92.122.34	132.92.36.122	UGHD	0	8543	ed0	
132.92.121.4	132.92.36.122	UGHD	0	0	ed0	
132.92.36.96/27	link#1	UC	0	0		
132.92.36.107	0:80:ad:17:96:34	UHLW	0	2385	ed0	1050
132.92.36.108	0:20:4c:30:29:29	UHLW	1	543	ed0	549
132.92.36.114	0:80:ad:a7:96:f5	UHLW	1	47862	lo0	
132.92.36.127	ff:ff:ff:ff:ff:ff	UHLWb	0	1	ed0	

Melihat status interface

```
mehyt# netstat –ni
```

Status interface yang dapat diketahui antara lain seperti di bawah ini.

```
Name  Mtu   Network       Address           Ipkts Ierrs    Opkts
ed0   1500  <Link>        00.80.ad.a7.96.f5 1233508   520   953655
ed0   1500  132.92.36.96  132.92.36.108     1233508   520   953655
tun0  1500  <Link>                            64346   412    52522
tun0  1500  132.92.21.120 132.92.21.121       64346   412    52522
lo0   16384 <Link>                            54180     0    54180
lo0   16384 127           127.0.0.1           54180     0    54180
```

6.1.3. route

Perintah ini untuk mengubah tabel routing secara manual.

Contoh:

Membuat entri routing ke jaringan 132.92.122.0/27 melalui 132.92.121.21

```
mehyt# route add –net 132.92.122.0 –netmask 255.255.255.224
132.92.121.21
```

Menghapus entri routing ke jaringan 132.92.122.0/27

```
mehyt# route delete –net 132.92.122.0 –netmask 255.255.255.224
```

Menghapus seluruh entri tabel routing

```
mehyt# route flush
```

6.2. Routing Statik di Jaringan

Pada bab sebelum ini telah diperkenalkan konsep dasar routing dan bagaimana membentuk tabel routing dalam sebuah jaringan yang sederhana. Dalam bagian ini kita akan membentuk tabel routing di router-router dalam jaringan yang lebih besar secara manual. Kita akan mampu memiliki gambaran mendasar mengenai konsep routing dengan membentuk tabel routing di router-router secara manual.

Pertama-tama perhatikan sebagian kecil dari jaringan pada Gambar 6.1 yaitu yang terhubung oleh router-router *khnemu*, *khensu*, dan *osiris*. Jaringan kecil ini akan kita gunakan untuk menjelaskan dasar implementasi routing di jaringan secara manual sebelum kita membuat tabel routing di router-router lain. Router *khnemu* memiliki dua interface yang masing-masing terhubung ke jaringan 132.92.122.96/28 dan 132.92.122.0/27. Router *khensu* terhubung langsung ke jaringan 132.92.122.32/27 dan 132.92.122.0/27, dan *osiris* adalah satu-satunya router menghubungkan jaringan-jaringan tersebut dengan bagian jaringan yang lain.

Berdasarkan data-data di atas kita hendak membuat tabel routing di ketiga router tersebut agar semua jaringan dapat saling terhubung. Pertama-tama kita buat tabel yang berisi entri-entri routing yang perlu kita tambahkan ke masing-masing router. Pada router *khnemu* dan *khensu* perlu ditambahkan dua entri sedangkan pada *osiris* perlu ditambahkan tiga buah entri. Untuk menambah entri tabel routing digunakan perintah route add. Pada router *khensu* perlu diberikan perintah berikut:

```
route add default 132.92.122.1
route add –net 132.92.122.96 –netmask 255.255.255.240 132.92.122.2
```

Pada router *khnemu*, perintahnya adalah:

```
route add default 132.92.122.1
route add –net 132.92.122.32 –netmask 255.255.255.224 132.92.122.3
```

Dan pada router *osiris*, perintah yang diberikan adalah:

```
route add default 132.92.121.34
route add –net 132.92.122.96 –netmask 255.255.255.240 132.92.122.2
route add –net 132.92.122.32 –netmask 255.255.255.224 132.92.122.3
```

Tabel 6.1 Penambahan routing untuk jaringan pada Gambar 6.2

router	Tujuan	Gateway
khnemu	0.0.0.0	132.92.122.1
	132.92.122.32/27	132.92.122.3
khensu	0.0.0.0	132.92.122.1
	132.92.122.96/28	132.92.122.2
osiris	0.0.0.0	132.92.121.34
	132.92.122.32/27	132.92.122.3
	132.92.122.96/28	132.92.122.2

Tabel 6.2 Tabel routing osiris

Destination	Gateway	Flags	Netif
default	132.92.121.34	UGSc	tun0
127.0.0.1	127.0.0.1	UH	lo0
132.92.121.33	127.0.0.1	UH	lo0
132.92.121.34	132.92.121.33	UH	tun0
132.92.122.0/27	link#1	UC	
132.92.122.1	0:80:ad:a7:96:f5	UHLW	lo0
132.92.122.31	ff:ff:ff:ff:ff:ff	UHLWb	ed0
132.92.122.32/27	132.92.122.3	UGSc	ed0
132.92.122.96/28	132.92.122.2	UGSC	ed0

Tabel 6.3 Tabel routing khnemu

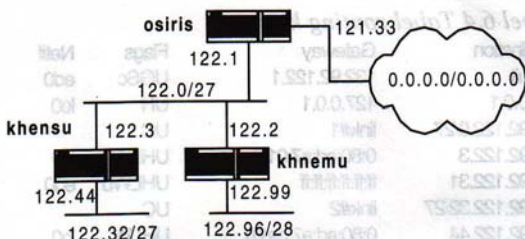
Destination	Gateway	Flags	Netif
default	132.92.122.1	UGSc	ed0
127.0.0.1	127.0.0.1	UH	lo0
132.92.122.0/27	link#1	UC	
132.92.122.2	0:80:ad:a7:c0:64	UHLW	lo0
132.92.122.31	ff:ff:ff:ff:ff:ff	UHLWb	ed0
132.92.122.32/27	132.92.122.3	UGSc	ed0
132.92.122.96/28	link#2	UC	
132.92.122.99	0:80:ad:a7:45:33	UHLW	lo0
132.92.122.111	ff:ff:ff:ff:ff:ff	UHLWb	ed1

Tabel 6.4 Tabel routing khensu

Destination	Gateway	Flags	Netif
default	132.92.122.1	UGSc	ed0
127.0.0.1	127.0.0.1	UH	lo0
132.92.122.0/27	link#1	UC	
132.92.122.3	0:80:ad:a7:01:3c	UHLW	lo0
132.92.122.31	ff:ff:ff:ff:ff:ff	UHLWb	ed0
132.92.122.32/27	link#2	UC	
132.92.122.44	0:80:ad:a7:ca:01	UHLW	lo0
132.92.122.63	ff:ff:ff:ff:ff:ff	UHLWb	ed1
132.92.122.96/28	132.92.122.2	UGSc	ed0

Parameter pada tabel routing FreeBSD di samping tujuan dan gateway adalah Flags dan Netif. Flags adalah parameter yang menunjukkan informasi mengenai rute tersebut sedangkan Netif adalah interface jaringan yang harus dilalui menuju rute tersebut. Beberapa flag pada tabel routing di atas antara lain:

Flags	Keterangan
b	Alamat broadcast
C	Rute sedang digunakan
c	Rute sedang digunakan, spesifik terhadap protokol
G	Rute perlu menggunakan gateway lagi
H	Rute ke host
L	Penerjemahan alamat link layer
S	Ditambah secara manual
W	Rute hasil cloning

Sekarang kita melihat bagian jaringan yang lain yang terdiri atas router-router *seth*, *amon*, *anubis*, dan *imsety* serta jaringan yang terhubung langsung dengan mereka. Dari topologi jaringan dapat kita ketahui bahwa rute default pada *seth*, *anubis*, dan *imsety* harus diarahkan ke *amon*. Router *seth* dengan jalur point-to-point ke *osiris* adalah satu-satunya jalur menuju jaringan yang dibicarakan di atas. Jadi, kita dapat menambahkan entri-entri berikut pada *seth*:

Gambar 6.2 Gambaran routing osiris, khnemu, khensu

Tujuan	Gateway
132.92.122.0/27	132.92.121.33
132.92.122.32/27	132.92.121.33
132.92.122.96/28	132.92.121.33

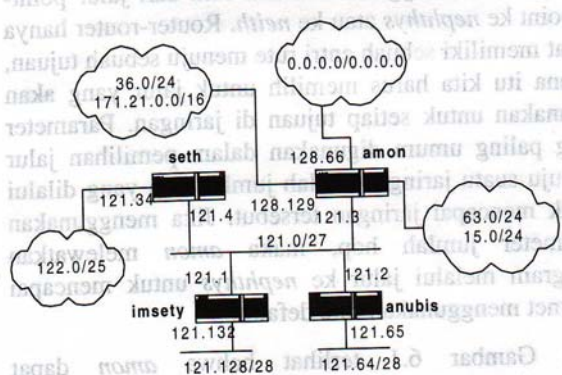
Seluruh rute di atas masuk terletak antara alamat IP 132.92.122.0 dan 132.92.122.128. Jaringan pada Gambar 6.1 tidak menggunakan alamat IP antara 132.92.122.0 dan 132.92.122.128 di bagian jaringan yang lain. Kondisi ini dapat dimanfaatkan dengan menggabungkan ketiga entri rute di atas menjadi satu entri, yaitu tujuan 132.92.122.0/25 melalui gateway 132.92.121.33. *Seth* juga harus diberikan entri routing untuk mencapai jaringan 132.92.121.128/28 melalui 132.92.121.1 dan 132.92.121.64/28 melalui 132.92.121.2. Dengan demikian, rute-rute yang perlu ditambahkan pada *seth* adalah:

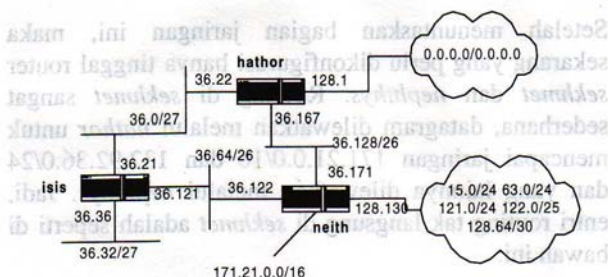
Tujuan	Gateway
0.0.0.0	132.92.121.3
132.92.121.128/28	132.92.121.1
132.92.121.64/28	132.92.121.2
132.92.122.0/25	132.92.121.33

Di router *amon*, tabel routing lebih rumit daripada di *seth*. *Amon* terhubung dengan dua buah jalur point-to-point, dengan jaringan 132.92.121.0/27, dan jaringan 132.92.15.0/24 dan 132.92.63.0/24 yang topologinya tidak digambarkan. Kedua rute terakhir dapat dicapai *amon* melalui 132.92.63.2. Selain itu *amon* juga harus dapat mencapai jaringan yang terhubung dengan *seth*, *anubis*, dan *imsety*.

Amon dapat mencapai jaringan yang terletak di sebelah atas gambar menggunakan salah satu dari jalur point-to-point ke *nephthys* atau ke *neith*. Router-router hanya dapat memiliki sebuah entri rute menuju sebuah tujuan, karena itu kita harus memilih untuk jalur yang akan digunakan untuk setiap tujuan di jaringan. Parameter yang paling umum digunakan dalam pemilihan jalur menuju suatu jaringan adalah jumlah hop yang dilalui untuk mencapai jaringan tersebut. Jika menggunakan parameter jumlah hop, maka *amon* melewatkan datagram melalui jalur ke *nephthys* untuk mencapai Internet menggunakan rute default.

Dari Gambar 6.1 terlihat bahwa *amon* dapat menggabungkan alamat beberapa jaringan yang dapat dicapai melalui *neith* menjadi sebuah alamat IP 132.92.36.0/24. *Amon* hendak menggunakan jalur ke *nephthys* untuk mencapai dua jalur point-to-point *nephthys-sekhmet* dan *sekhmet-hathor* (132.92.128/30) dan jaringan 132.92.129.0/27. Karena jalur *nephthys* digunakan sebagai rute default, maka *amon* tidak perlu mencantumkan entri routing jaringan-jaringan lain yang melalui *nephthys*. Gambar 6.3 memperlihatkan gambaran kasar routing di router-router *seth*, *amon*, *anubis*, dan *imsety*. Rute-rute yang perlu ditambahkan pada tabel routing di *amon* adalah:

Tujuan	Gateway
0.0.0.0	132.92.128.65
132.92.15.0/24	132.92.63.1
132.92.36.0/24	132.92.128.130
132.92.63.0/24	132.92.63.1
132.92.121.32/30	132.92.121.4
132.92.121.64/28	132.92.121.2
132.92.121.128/28	132.92.121.1
132.92.122.0/25	132.92.121.4
171.21.0.0/16	132.92.128.130

Gambar 6.3. Gambaran routing seth, amon, anubis, *dan* imsety

Pada bagian jaringan yang terhubung dengan router-router *hathor*, *isis*, dan *neith* juga terdapat dua jalur yang dapat digunakan untuk menuju Internet, yaitu jalur point-to-point *neith-amon* dan *hathor-sekhmet*. Rute default yang kita pilih untuk implementasi routing di jaringan ini adalah rute melalui *hathor*. Gambaran routing untuk jaringan di bagian ini adalah seperti yang terlihat pada Gambar 6.4.

Gambar 6.4. Gambaran routing hathor, isis, *dan* neith

Dengan gambaran routing tersebut, entri tabel routing yang perlu ditambahkan di *hathor* adalah:

Tujuan	Gateway
0.0.0.0	132.92.128.2
132.92.15.0/24	132.92.36.171
132.92.36.32/27	132.92.36.21
132.92.36.64/26	132.92.36.21
132.92.63.0/24	132.92.36.171
132.92.121.0/24	132.92.36.171
132.92.122.0/25	132.92.36.171
132.92.128.64/30	132.92.36.171
171.21.0.0/16	132.92.36.171

Entri routing tak langsung di *neith* adalah sebagai berikut:

Tujuan	Gateway
0.0.0.0	132.92.36.167
132.92.15.0/24	132.92.128.29
132.92.36.0/26	132.92.36.21
132.92.63.0/24	132.92.128.29
132.92.121.0/24	132.92.128.29
132.92.122.0/25	132.92.128.29
132.92.128.64/30	132.92.128.29
171.21.0.0/16	171.21.13.8

Setelah menuntaskan bagian jaringan ini, maka sekarang yang perlu dikonfigurasi hanya tinggal router *sekhmet* dan *nephthys*. Routing di *sekhmet* sangat sederhana, datagram dilewatkan melalui *hathor* untuk mencapai jaringan 171.21.0.0/16 dan 132.92.36.0/24 dan yang lainnya dilewatkan melalui *nephthys*. Jadi, entri routing tak langsung di *sekhmet* adalah seperti di bawah ini:

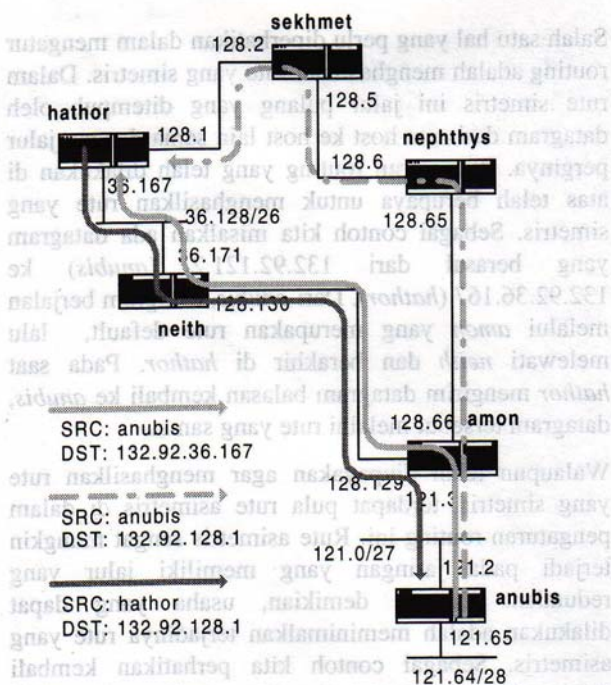
Tujuan	Gateway
0.0.0.0	132.92.128.6
132.92.36.0/24	132.92.128.1
171.21.0.0/16	132.92.128.1

Router *nephthys* merupakan satu-satunya router yang menghubungkan jaringan 132.92.0.0/16 dan 171.21.0.0/16 dengan Internet. Kita asumsikan bahwa *nephthys* mencapai Internet melalui 167.205.23.1. Dengan demikian router *nephthys* juga menjadi jalan masuk bagi setiap datagram yang menuju kedua jaringan tersebut. Konsekuensinya adalah *nephthys* harus mengetahui jalur-jalur untuk mencapai seluruh jaringan di dalam 132.92.0.0/16 dan 171.21.0.0/16 sebab rute default di *nephthys* mengarah ke luar kedua jaringan tersebut. Di *nephthys* kita juga melakukan penggabungan rute seperti pada router-router lain. Entri routing yang harus ditambahkan di *nephthys* adalah seperti berikut:

Tujuan	Gateway
0.0.0.0	167.205.23.1
132.92.15.0/24	132.92.15.67
132.92.36.0/24	132.92.128.5
132.92.63.0/24	132.92.128.66
132.92.121.0/24	132.92.128.66
132.92.122.0/25	132.92.128.66
132.92.128.0/30	132.92.128.5
132.92.129.0/27	132.92.128.5
171.21.0.0/16	132.92.128.5

Salah satu hal yang perlu diperhatikan dalam mengatur routing adalah menghasilkan rute yang simetris. Dalam rute simetris ini jalur pulang yang ditempuh oleh datagram dari satu host ke host lain sama dengan jalur perginya. Pengaturan routing yang telah dilakukan di atas telah berupaya untuk menghasilkan rute yang simetris. Sebagai contoh kita misalkan ada datagram yang berasal dari 132.92.121.2 (*anubis*) ke 132.92.36.167 (*hathor*). Dari *anubis*, datagram berjalan melalui *amon* yang merupakan rute default, lalu melewati *neith* dan berakhir di *hathor*. Pada saat *hathor* mengirim datagram balasan kembali ke *anubis*, datagram tersebut melalui rute yang sama.

Walaupun telah diupayakan agar menghasilkan rute yang simetris, terdapat pula rute asimetris di dalam pengaturan routing ini. Rute asimetris sangat mungkin terjadi pada jaringan yang memiliki jalur yang redundan. Dengan demikian, usaha yang dapat dilakukan adalah meminimalkan terjadinya rute yang asimetris. Sebagai contoh kita perhatikan kembali routing dari *anubis* ke *hathor* dan sebaliknya, tetapi IP *hathor* yang digunakan sekarang adalah 132.92.128.1. Dari *anubis* datagram dilewatkan melalui *amon*. Tabel

routing di *amon* menunjukkan bahwa untuk mencapai 132.92.128.1 datagram harus dilewatkan melalui *nephthys*. Oleh *nephthys* datagram dilewatkan melalui *sekhmet* sebelum mencapai *hathor*. Ketika mengirimkan datagram balasan, *hathor* melewatkan datagram melalui *neith* yang selanjutnya dilewatkan melalui *amon* sebelum sampai ke *anubis*. Rute seperti inilah yang disebut sebagai rute asimetris, dan divisualisasikan di Gambar 6.5.

Gambar 6.5. Routing simetris dan asimetris antara anubis *dan* hathor

Dari pembentukan routing jaringan secara manual ini kita dapat memperoleh gambaran bagaimana router-router dalam jaringan melewatkan datagram untuk mencapai tujuan.

Pada bagian berikut kita akan menjalankan protokol routing menggunakan daemon routing GateD di FreeBSD. Pertama-tama kita akan melihat GateD secara sepintas beserta perintah-perintahnya, kemudian kita akan menggunakan GateD untuk menjalankan RIPv2 di jaringan. Terakhir, kita akan menggunakan GateD untuk menjalankan protokol routing OSPF.

6.3. GateD (Gate Daemon)

GateD adalah daemon routing yang dikembangkan oleh Merit GateD Consortium. GateD dapat menjalankan bermacam-macam protokol routing, seperti RIP, Hello, OSPF, EGP, dan BGP. Selain itu GateD juga dapat mengekspor rute dari satu protokol routing ke protokol routing yang lain. Saat ini GateD digunakan di banyak jaringan di Internet serta menjadi kode dasar dari beberapa produk router. GateD yang digunakan dalam implementasi ini adalah versi 3.5 yang dapat diambil melalui **http://www.gated.org**.

GateD dapat dijalankan di banyak versi Unix. GateD dapat dijalankan secara langsung atau menggunakan gdc, yaitu program interface gated ke pengguna. Pada saat mulai dijalankan, GateD membaca file /etc/gated.conf yang berisi konfigurasi routing. GateD dapat diberi sinyal-sinyal untuk membaca kembali file konfigurasi, menyimpan status ke file, dan beberapa aktivitas lain.

6.3.1. Perintah konfigurasi GateD

Sebelum menggunakan GateD di jaringan, kita akan melihat beberapa perintah konfigurasi yang diperlukan untuk menjalankan protokol routing RIP dan OSPF. Penjelasan di buku ini berdasarkan dokumentasi GateD dan tidak dimaksudkan untuk menjelaskan konfigurasi GateD secara menyeluruh. Pembaca yang hendak mengetahui perintah-perintah konfigurasi GateD dapat membaca dokumentasi yang menjadi bagian dari distribusi GateD.

Konvensi penulisan yang digunakan di bagian ini sama seperti dokumentasi GateD, yaitu perintah-perintah menggunakan huruf tebal; parameter menggunakan huruf miring. Parameter yang terletak di dalam tanda kurung siku ('[' dan ']') bersifat opsional. Garis vertikal '|' digunakan untuk menyatakan pilihan parameter; tanda kurung '(',')' digunakan untuk mengelompokkan parameter jika diperlukan.

Contoh:

[**backbone** | (**area** *area*)]

pada baris di atas kita dapat memilih menggunakan **backbone** atau **area**. Jika menggunakan **area**, maka harus memasukkan parameter *area*.

Berikut ini adalah perintah-perintah konfigurasi GateD yang akan digunakan dalam buku ini.

routerid	Untuk mendefinisikan identitas router, digunakan pada protokol OSPF dan BGP.
rip	Untuk mengkonfigurasi protokol RIP.
ospf	Untuk mengkonfigurasi protokol OSPF.

static Untuk memberikan entri routing statik pada router.

export Untuk mengatur ekspor rute dari satu protokol ke protokol lain.

Penjelasan mengenai perintah-perintah tersebut dapat dilihat di bawah ini.

6.3.2. Konfigurasi Router ID

```
routerid host ;
```

Perintah ini menentukan identitas router jika menjalankan protokol BGP atau OSPF. Secara default GateD menentukan identitas router dari interface pertama yang ditemukan GateD.

6.3.3. Konfigurasi RIP

```
rip yes | no | on | off [ {
  broadcast ;
  nobroadcast ;
  nocheckzero ;
  preference preference ;
  defaultmetric metric ;
  query authentication [none | [[simple|md5] password]] ;
  interface interface_list
    [noripin] | [ripin]
    [noripout] | [ripout]
    [metricin metric]
    [metricout metric]
    [version 1]|[version 2 [multicast|broadcast]]
   [[secondary] authentication [none | [[simple|md5] password]] ;
  trustedgateways gateway_list ;
  sourcegateways gateway_list ;
  traceoptions trace_options ;
} ] ;
```

Perintah rip digunakan untuk mengaktifkan dan mematikan protokol routing RIP. Secara default GateD menyalakan protokol RIP walaupun tidak didefinisikan dalam file konfigurasi. Jika terdapat hanya satu interface, RIP akan berjalan dengan mode nobroadcast dan berjalan dengan mode `broadcast` jika terdapat lebih dari satu interface

Pilihan-pilihan untuk perintah rip antara lain sebagai berikut:

broadcast

Host mengirimkan paket RIP ke jaringan, walaupun hanya terdapat sebuah interface saja.

nobroadcast

Host hanya mendengar paket RIP (menjadi silent node), walaupun terdapat lebih dari satu interface.

interface interface_list

Mengatur parameter-parameter untuk interface yang terdapat dalam *interface_list*. *Interface_list* dapat berupa nama atau alamat IP interface. Parameter yang sering digunakan untuk interface adalah:

noripin

mengatur agar tidak menghiraukan paket RIP yang diterima, defaultnya adalah **ripin**.

noripout

mengatur agar tidak mengirimkan paket RIP, defaultnya adalah ripout.

version 1

mengatur agar paket RIP yang dikirim adalah RIP versi 1. Ini adalah mode default

version 2

mengatur agar paket RIP yang dikirim adalah RIP versi 2.

multicast

RIP versi 2 dikirimkan melalui interface menggunakan multicast. Ini adalah mode default jika menggunakan RIP versi 2.

broadcast

RIP versi 2 dikirimkan secara broadcast dan kompatibel dengan RIP versi 1

6.3.4. Konfigurasi OSPF

```
ospf yes | no | on | off [ {
  defaults {
    preference preference ;
    cost cost ;
    tag [ as ] tag ;
    type 1 | 2 ;
  } ;
  exportlimit routes ;
  exportinterval time ;
  traceoptions trace_options ;
  monitorauthkey authkey ;
  monitorauth none | ( [ simple | md5 ] authkey ) ;
  backbone | ( area area ) {
    authtype 0 | 1 | none | simple ;
    stub [ cost cost] ;
    networks {
      network [ restrict ] ;
      network mask mask [ restrict ] ;
      network masklen number [ restrict ] ;
      host host [ restrict ] ;
    } ;
    stubhosts {
      host cost cost ;
    } ;
    interface interface_list; [cost cost ] {
```

```
            interface_parameters
        };
        interface interface_list nonbroadcast [cost cost ] {
            pollinterval time ;
            routers {
                gateway [ eligible ] ;
            };
            interface_parameters
        };
        Backbone only:
        virtuallink neighborid router_id transitarea area {
            interface_parameters
        };
    };
}];
```

Atribut-atribut pada *interface_parameters* dalam perintah interface adalah seperti di bawah ini.

```
enable | disable ;
retransmitinterval time ;
transitdelay time ;
priority priority ;
hellointerval time ;
routerdeadinterval time ;
authkey auth_key ;
```

Pilihan-pilihan yang umum digunakan dalam mengkonfigurasi protokol OSPF adalah sebagai berikut:

backbone

area area

Setiap router OSPF harus terletak dalam sebuah area. Jika router terletak di backbone, klausa yang digunakan adalah backbone dan bukan area 0.

authtype 0 | 1 | none | simple

Menentukan skim autentikasi di area. Skim di sebuah area harus sama, tetapi autentikasi setiap jaringan boleh berbeda.

stub [cost cost]

menentukan area adalah stub sehingga tidak ada ASE yang dimasukkan ke area ini kecuali rute default dengan biaya *cost*.

networks

Menentukan jaringan-jaringan yang masuk dalam area. Jika restrict didefinisikan jaringan tersebut tidak akan dimasukkan ke dalam paket LSA.

stubhosts

Menentukan host-host yang terhubung langsung dengan router

interface interface_list [cost cost]

klausa ini untuk mengkonfigurasi interface broadcast atau point-to-point dengan biaya interface yang didefinisikan dengan **cost**. Default biaya ini adalah satu.

Parameter interface adalah:

retransmitinterval *time*

selang waktu dalam detik antara setiap pengiriman paket LSA

transitdelay *time*

menentukan perkiraan delay pada saat mengirimkan paket LSA

priority *priority*
bilangan antara 0 sampai dengan 255 untuk menentukan pemilihan DR di jaringan broadcast. Router dengan priority 0 tidak dapat menjadi DR.

hellointerval *time*
selang waktu pengiriman paket Hello

routerdeadinterval *time*
selang waktu sampai router dinyatakan mati

authkey *auth_key*
kunci untuk mengirim dan menerima paket OSPF

6.3.5. Konfigurasi routing statik

```
static {
   ( host host ) | default |
   ( network [ ( mask mask ) | ( masklen number ) ] )
      gateway gateway_list
      [ interface interface_list ]
      [ preference preference ]
      [ retain ]
      [ reject ]
      [ blackhole ]
      [ noinstall ] ;
( network [ ( mask mask ) | ( masklen number ) ] )
      [ interface interface_list ]
      [ preference preference ]
      [ retain ]
      [ reject ]
      [ blackhole ]
      [ noinstall ] ;
} ;
```

GateD juga dapat memasukkan sejumlah routing statik ke tabel routing.

(**host** *host*) | **default** |
(**network** [(**mask** *mask*) | (**masklen** *number*)])
gateway *gateway_list*

Bentuk seperti di atas adalah bentuk routing statik yang paling sederhana.
Parameter-parameter pada routing statik adalah:

interface *interface_list*
jika parameter ini muncul, gateway dianggap valid hanya jika dapat dicapai melalui *interface_list*

preference *preference*
menentukan preference atas routing statik

retain
rute ini tidak dihapuskan dari kernel walaupun GateD dimatikan

reject
datagram menuju rute ini dibuang dan dinyatakan tidak dapat dicapai

blackhole
mirip dengan reject

noinstall
rute tidak masukkan ke dalam tabel routing kernel

6.3.6. Konfigurasi penyebaran rute melalui export

Sintaks penyebaran rute melalui export tergantung protokol routing yang akan digunakan. Di buku ini kita hanya akan memperhatikan export ke RIP dan OSPF dari protokol routing lain dan routing statik.

Mengekspor rute ke RIP dan HELLO

```
export proto rip | hello
   [ ( interface interface_list ) | (gateway gateway_list ) ]
   restrict ;
export proto rip | hello
   [ ( interface interface_list ) | (gateway gateway_list ) ]
   [ metric metric ] {
   export_list ;
} ;
```

Perintah ini untuk mengekspor rute dalam *export_list* ke protokol RIP pada interface *inteface_list* atau *gateway_list* dengan metrik *metric*. Jika **restrict** terdefinisi, maka ekspor ke protokol RIP tidak diperbolehkan.

Mengekspor rute ke OSPF

```
export proto osfpase [ type 1 | 2 ] [ tag ospf_tag ]
   restrict ;
export proto osfpase [ type 1 | 2 ] [ tag ospf_tag ]
   [ metric metric ] {
   export_list ;
} ;
```

Protokol OSPF hanya membolehkan rute eksternal dimasukkan ke OSPF dalam bentuk OSPF ASE, tipe 1 atau 2. Setiap OSPF ASE dapat memiliki tag yang digunakan untuk memfilter rute. Perintah di atas mengekspor rute dalam *export_list* ke protokol OSPF dalam bentuk ASE tipe 1 atau tipe 2.

6.3.7. Sumber rute yang hendak diekspor

Export list mendefinisikan sumber-sumber rute yang hendak diekspor. Sumber ini dapat berasal dari

protokol routing atau sumber lain, misal dari interface. Di bawah ini diperlihatkan beberapa sumber rute yang dapat diekspor. Rute-rute yang diekspor ke protokol lain adalah rute yang masuk dalam *route_filter*.

Protokol RIP atau HELLO

```
proto rip | hello
    [ ( interface interface_list ) | (gateway gateway_list ) ]
    restrict ;
proto rip | hello
    [ ( interface interface_list ) | (gateway gateway_list ) ]
    [ metric metric ] {
    route_filter [ restrict | ( metric metric ) ] ;
} ;
```

Protokol OSPF

```
proto ospf | ospfase restrict ;
proto ospf | ospfase [ metric metric ] {
    route_filter [ restrict | ( metric metric ) ] ;
} ;
```

Sumber selain protokol routing

```
proto direct | static | kernel
    [ (interface interface_list ) ]
    restrict ;
proto direct | static | kernel
    [ (interface interface_list ) ]
    [ metric metric ] {
    route_filter [ restrict | ( metric metric ) ] ;
} ;
```

Sumber selain protokol routing antara lain adalah:

direct

Rute yang langsung terhubung dengan interface

static

Rute yang didefinisikan dalam perintah static

kernel

Rute yang dimasukkan secara manual ke dalam kernel menggunakan perintah 'route'

6.3.8. Memfilter rute

Format untuk memfilter rute adalah seperti di bawah ini. Rute yang akan diekspor ke protokol lain hanyalah rute-rute yang tertulis di dalam *route_filter*. GateD mendefinisikan bahwa di akhir *route_filter* terdapat baris yang berisi `all restrict ;` walaupun tidak tertulis dalam file konfigurasi.

network [**exact** | **refines**]
network **mask** *mask* [**exact** | **refines**]
network **masklen** *number* [**exact** | **refines**]
default
host *host*

6.4. Routing Menggunakan RIPv2

Kita telah melihat sepintas perintah-perintah konfigurasi GateD yang akan digunakan pada jaringan Gambar 6.1, dan sekarang kita akan membuat konfigurasi GateD yang akan dimasukkan dalam /etc/gated.conf.

Ketika hendak menjalankan protokol routing RIP di suatu jaringan, kita harus memperhatikan dahulu apakah jaringan tersebut menggunakan subnet mask yang bervariasi. Jika ya, maka RIPv1 tidak dapat digunakan di jaringan tersebut. Jaringan pada Gambar 6.1 menggunakan subnet mask yang bervariasi sehingga jika hendak menggunakan routing RIP pada jaringan tersebut maka harus menggunakan RIP versi 2.

Konfigurasi routing RIP di GateD sangat sederhana. Untuk memperlihatkan konfigurasi routing RIP versi 2 di jaringan, kita akan memeriksa konfigurasi GateD beberapa router, yaitu: *isis*, *neith*, *amon*, dan *nephthys*. Konfigurasi GateD masing-masing router dapat dilihat pada Gambar 6.6 sampai 6.9.

```
rip on {
    broadcast ;
    interface 132.92.36.21 ripin ripout version 2 multicast ;
    interface 132.92.36.36 ripin ripout version 2 multicast ;
    interface 132.92.36.121 ripin ripout version 2 multicast ;
} ;
```

Gambar 6.6 Konfigurasi GateD di router isis

```
rip yes {
    broadcast ;
    interface 132.92.36.122 ripin ripout version 2 multicast ;
    interface 132.92.36.171 ripin ripout version 2 multicast ;
    interface 132.92.128.130 ripin ripout version 2 multicast ;
} ;
static {
    171.21 masklen 16 gateway 171.21.13.8 ;
} ;
export proto rip {
    proto rip {
        all ;
    } ;
    proto direct {
        all ;
    } ;
    proto static {
        171.21 masklen 16 exact metric 2 ;
    } ;
} ;
```

Gambar 6.7 Konfigurasi GateD di router neith

```
rip yes {
    broadcast ;
    interface 132.92.121.3 ripin ripout version 2 multicast ;
    interface 132.92.128.66 ripin ripout version 2 multicast ;
    interface 132.92.128.129 ripin ripout version 2 multicast ;
} ;
static {
    132.92.15 masklen 24 gateway 132.92.63.1 ;
    132.92.63.0 mask 255.255.255.0 gateway 132.92.63.1 ;
} ;
export proto rip {
    proto rip {
        all ;
    } ;
    proto direct {
        all ;
    } ;
    proto static {
        all metric 2 ;
    } ;
} ;
```

Gambar 6.8 Konfigurasi GateD di router amon

```
rip yes {
    broadcast ;
    interface 132.92.128.6 ripin ripout version 2 multicast ;
    interface 132.92.128.65 ripin ripout version 2 multicast ;
} ;
static {
    default gateway 167.205.23.1 ;
    132.92.15 masklen 24 gateway 132.92.15.67 ;
} ;
export proto rip {
    proto rip {
        all ;
    } ;
    proto direct {
        all ;
    } ;
    proto static {
```

```
            132.92.0.0 mask 255.255.0.0 refines metric 1 ;
            default metric 3 ;
        };
};
```

Gambar 6.9 Konfigurasi GateD di router nephthys

Router *isis* menjalankan RIPv2 di seluruh interface yang dimilikinya dan dapat menerima dan mengirimkan paket RIP ke seluruh interface tersebut. Pada baris pertama konfigurasi router *isis* dituliskan `rip on` kemudian diikuti parameter-parameter untuk protokol RIP. Parameter setiap interface menunjukkan bahwa setiap interface dapat menerima dan mengirim paket RIP dalam format versi 2 secara multicast dengan metrik default: satu. Router-router lain dalam jaringan juga diatur seperti router *isis*, yaitu menggunakan paket RIPv2 multicast di semua interface-nya.

Pada konfigurasi router *neith*, *amon*, dan *nephthys* ketiganya menggunakan routing statik untuk memasukkan entri ke dalam tabel routing. Karena informasi routing statik tersebut juga harus disampaikan kepada router-router lain maka informasi dalam routing statik tersebut diekspor ke protokol RIP menggunakan pernyataan `export proto rip`. Pernyataan untuk mengekspor ke protokol RIP mensyaratkan bahwa informasi dari protokol RIP dan jaringan yang terhubung langsung juga harus dinyatakan secara eksplisit. Itulah sebab mengapa terdapat baris proto rip dan proto direct, selain proto static.

Ketiga router yang mengekpor routing statik ke protokol RIP menggunakan filter rute dan metrik yang berbeda. *Neith* hanya mengekspor rute menuju 171.21.0.0/16 saja dengan metrik 2. Jadi jika terdapat

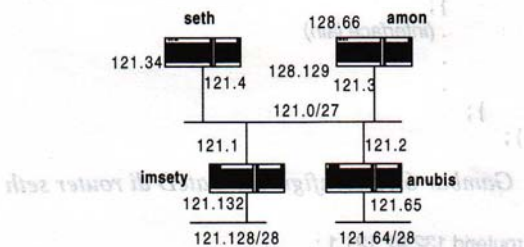
rute-rute lain di bagian routing statik tidak akan diekspor ke protokol RIP. *Amon* mengekspor semua rute yang terdapat di bagian routing statik dengan metrik 2 sehingga rute apa pun yang dimasukkan di bagian routing statik akan muncul dalam protokol RIP. Sementara itu *nephthys* mengekspor rute ke jaringan-jaringan dalam jangkauan 132.92.0.0/16 dengan metrik 1 dan juga mengekspor rute default dengan metrik 3. Dengan demikian, router-router di jaringan sekarang memiliki informasi routing ke seluruh jaringan 132.92.0.0/16, ke jaringan 171.21.0.0/16 dan ke Internet menggunakan rute default.

6.5. Routing Menggunakan OSPF

Dalam menjelaskan teknik mengkonfigurasi routing OSPF dengan GateD, kita akan mulai dengan melihat sebuah jaringan broadcast untuk mengatur pemilihan Designated Router dan Backup Designated Router di jaringan tersebut. Setelah itu kita melihat konfigurasi jaringan OSPF tanpa area dan dengan membaginya menjadi area.

6.5.1. Prioritas Interface dan Designated Router

Perhatikan jaringan 132.92.121.0/27 dan router-router yang terhubung dengannya: *seth*, *amon*, *anubis*, dan *imsety* (Gambar 6.10). Kita menginginkan agar urutan prioritas menjadi Designated Router di jaringan ini adalah *anubis*, *seth*, *imsety*, dan *amon*. Dengan demikian kita harus mengatur parameter priority di konfigurasi GateD yang mencermin urutan prioritas terse-

but. Biaya interface jaringan broadcast ini ditetapkan sebesar 10. Pada jaringan ini diinginkan selang waktu antar paket Hello selama 10 detik dan jika sebuah router tidak terdengar selama 40 detik router tersebut dianggap mati. Selain itu setiap paket LSA diautentikasi menggunakan tipe *simple* dengan kunci *testOSPF*. Konfigurasi GateD router *anubis*, *seth*, *imsety*, dan *amon* untuk interface yang terhubung dengan jaringan 132.92.121.0/27 berturut-turut dapat dilihat pada Gambar 6.11 sampai dengan Gambar 6.14.

Gambar 6.10 Jaringan 13.92.12.0/27 dengan 4 router

```
routerid 132.92.121.2 ;
ospf yes {
    backbone {
        authtype simple ;
        interface 132.92.121.2 cost 10 {
                                    priority 250 ;
                                    hellointerval 10 ;
                                    routerdeadinterval 40 ;
                                    authkey "testOSPF" ;
        } ;
        . (interface lain)
        .
        .
    } ;
} ;
```

Gambar 6.11 Konfigurasi GateD di router anubis

```
routerid 132.92.121.4 ;
ospf yes {
    backbone {
        authtype simple ;
        interface 132.92.121.4 cost 10 {
                                priority 200 ;
                                hellointerval 10 ;
                                routerdeadinterval 40 ;
                                authkey "testOSPF" ;
        };
        . (interface lain)
        .
        .
    };
};
```

Gambar 6.12 Konfigurasi GateD di router seth

```
routerid 132.92.121.1 ;
ospf yes {
    backbone {
        authtype simple ;
        interface 132.92.121.1 cost 10 {
                                priority 150 ;
                                hellointerval 10 ;
                                routerdeadinterval 40 ;
                                authkey "testOSPF" ;
        };
        . (interface lain)
        .
        .
    };
};
```

Gambar 6.13 Konfigurasi GateD di router imsety

```
routerid 132.92.121.3 ;
ospf yes {
```

```
backbone {
    authtype simple ;
    interface 132.92.121.3 cost 10 {
                            priority 100 ;
                            hellointerval 10 ;
                            routerdeadinterval 40 ;
                            authkey "testOSPF" ;
    } ;
    . (interface lain)
    .
    .
  } ;
} ;
```

Gambar 6.14 Konfigurasi GateD di router amon

Dengan konfigurasi router-router seperti di atas, maka setiap 30 detik terjadi pertukaran paket Hello antar router. Ketika semua router baru menyala dan belum ada Designated Router dan Backup Designated Router di jaringan tersebut, router-router memilih DR dan BDR. Setelah pertukaran paket Hello, router-router tersebut memilih *anubis* menjadi DR dan *seth* menjadi BDR. Jika pada suatu saat *anubis* dianggap mati karena tidak terdengar selama 120 detik, router-router kembali memilih BDR. Router *seth* yang sekarang menjadi BDR dipromosikan menjadi DR dan BDR dipilih dari router *imsety* dan *amon*. Karena *imsety* memiliki prioritas yang lebih besar, maka *imsety* terpilih menjadi BDR.

6.5.2. Jaringan OSPF tanpa Area

Sekarang kita melihat konfigurasi GateD jika jaringan tidak dibagi menjadi beberapa area. Parameter-parameter yang kita tentukan untuk jaringan OSPF ini adalah:

- Jaringan broadcast
- cost interface 10
- interval hello 10 detik
- selang sebelum router dianggap mati 40 detik
- kunci autentikasi *lanospf*
- Jaringan point-to-point
- Cost interface 50
- Interval hello 120 detik
- selang sebelum router dianggap mati 1200 detik
- interval pengiriman kembali LSA 10 detik
- kunci autentikasi *wanospf*
- Ekspor routing statik ke OSPF
- Cost 50
- ASE tipe 1

Untuk memperlihatkan konfigurasi GateD di jaringan ini akan diberikan contoh konfigurasi empat buah router: *khnemu*, *neith*, *amon*, dan *nephthys*. Router *khnemu* digunakan sebagai model untuk konfigurasi router-router di jaringan yang seluruh interface-nya terhubung ke jaringan broadcast. Router-router *neith*, *amon*, dan *nephthys* memperlihatkan konfigurasi jaringan dengan interface point-to-point dan ekspor routing statik ke OSPF. Konfigurasi router-router di atas diperlihatkan oleh Gambar 6.15 sampai dengan Gambar 6.18.

```
routerid 132.92.122.2 ;
rip no ;
ospf yes {
    backbone {
        authtype simple ;
```

```
        interface 132.92.122.2 cost 10 {
                                priority 100 ;
                                hellointerval 10 ;
                                routerdeadinterval 40 ;
                                authkey "lanospf" ;
        };
        interface 132.92.122.99 cost 10 {
                                priority 250 ;
                                hellointerval 10 ;
                                routerdeadinterval 40 ;
                                authkey "lanospf" ;
        };
    };
};
```

Gambar 6.15 Konfigurasi GateD di khnemu

```
routerid 132.92.36.122 ;
rip no ;
ospf yes {
    backbone {
        authtype simple ;
        interface 132.92.36.122 cost 10 {
                                priority 100 ;
                                hellointerval 10 ;
                                routerdeadinterval 40 ;
                                authkey "lanospf" ;
        };
        interface 132.92.36.171 cost 10 {
                                priority 250 ;
                                hellointerval 10 ;
                                routerdeadinterval 40 ;
                                authkey "lanospf" ;
        };
        # jaringan point-to-point
        interface 132.92.128.130 cost 50 {
                                priority 250 ;
                                retransmitintreval 10 ;
                                hellointerval 120 ;
                                routerdeadinterval 1200 ;
```

```
                                   authkey "wanospf" ;
        } ;
    } ;
} ;
static {
    171.21 masklen 16 gateway 171.21.13.8 ;
} ;
export proto ospfase type 1 {
    proto static {
        171.21 masklen 16 exact metric 50 ;
    } ;
};
```

Gambar 6.16 Konfigurasi GateD di neith

```
routerid 132.92.121.3 ;
rip no ;
ospf yes {
    backbone {
        authtype simple ;
        interface 132.92.121.3 cost 10 {
                                   priority 100 ;
                                   hellointerval 10 ;
                                   routerdeadinterval 40 ;
                                   authkey "lanospf" ;
        } ;
        # jaringan point-to-point
        interface 132.92.128.66 cost 50 {
                                   priority 150 ;
                                   retransmitintreval 10 ;
                                   hellointerval 120 ;
                                   routerdeadinterval 1200 ;
                                   authkey "wanospf" ;
        } ;
        interface 132.92.128.129 cost 50 {
                                   priority 150 ;
                                   retransmitintreval 10 ;
                                   hellointerval 120 ;
                                   routerdeadinterval 1200 ;
                                   authkey "wanospf" ;
```

```
        };
    };
};
static {
    132.92.15 masklen 24 gateway 132.92.63.1 ;
    132.92.63.0 mask 255.255.255.0 gateway 132.92.63.1 ;
};
export proto ospfase type 1 {
    proto static {
        all metric 50 ;
    };
};
```

Gambar 6.17 Konfigurasi GateD di amon

```
routerid 132.92.128.6 ;
rip no ;
ospf yes {
    backbone {
        authtype simple ;
        # jaringan point-to-point
        interface 132.92.128.6 cost 50 {
                        priority 250 ;
                        retransmitintreval 10 ;
                        hellointerval 120 ;
                        routerdeadinterval 1200 ;
                        authkey "wanospf" ;
        };
        interface 132.92.128.65 cost 50 {
                        priority 250 ;
                        retransmitintreval 10 ;
                        hellointerval 120 ;
                        routerdeadinterval 1200 ;
                        authkey "wanospf" ;
        };
    };
};
static {
    default gateway 167.205.23.1;
    132.92.15 masklen 24 gateway 132.92.15.67 ;
```

```
};
export proto ospfase type 1 {
    proto static {
        all metric 50 ;
    };
};
```

Gambar 6.18 Konfigurasi GateD di nephthys

Konfigurasi GateD untuk protokol OSPF lebih rumit daripada konfigurasi untuk RIP. Seperti terlihat pada gambar-gambar di atas, konfigurasi untuk OSPF memerlukan lebih banyak jumlah baris daripada konfigurasi RIP.

Dalam mengatur parameter-parameter interface terdapat beberapa hal yang harus diperhatikan. Pertama adalah penentuan biaya (cost) interface. Konvensi cost interface yang digunakan OSPF adalah

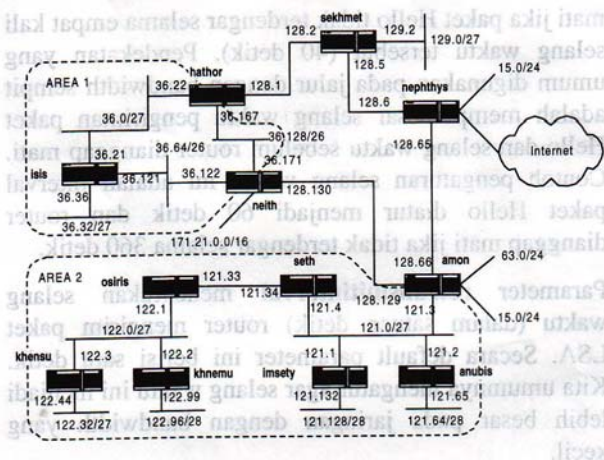
cost = 100.000.000 / bandwidth dalam bps. Penentuan cost berdasarkan kecepatan ini cukup wajar karena berhubungan dengan masalah kejenuhan jalur. Misal untuk jalur Ethernet cost interface-nya adalah 100.000.000/ 10.000.000 = 10, dan untuk jalur T1 (1.544Mbps) cost interface-nya adalah 64.

Parameter **hellointerval** dan **routerdeadinterval** mengatur waktu antar paket Hello dan selang waktu sampai router dianggap mati. Parameter ini juga berhubungan dengan kecepatan jalur. Kita tidak ingin router terlalu sering mengirimkan paket Hello ke jalur berkecepatan rendah karena hanya menghabiskan bandwidth. Sebagai aturan umum kita menginginkan interval paket Hello yang lebih lama jika bandwidth jalur sempit. Secara default interval paket Hello di GateD adalah 10 detik dan sebuah router dianggap

mati jika paket Hello tidak terdengar selama empat kali selang waktu tersebut (40 detik). Pendekatan yang umum digunakan pada jalur dengan bandwidth sempit adalah memperbesar selang waktu pengiriman paket Hello dan selang waktu sebelum router dianggap mati. Contoh pengaturan selang waktu ini adalah interval paket Hello diatur menjadi 60 detik dan router dianggap mati jika tidak terdengar selama 360 detik.

Parameter **retransmitinterval** menentukan selang waktu (dalam satuan detik) router mengirim paket LSA. Secara default parameter ini berisi satu detik. Kita umumnya mengatur agar selang waktu ini menjadi lebih besar pada jaringan dengan bandwidth yang kecil.

6.5.3. Jaringan OSPF dengan Area

Kembali kita perhatikan jaringan pada Gambar 6.1 yang kali ini dibagi menjadi dua area OSPF seperti pada Gambar 6.19. Area 1 meliputi jaringan 132.92.36.0/24 dengan dua router perbatasan area: *neith* dan *hathor*. Area 2 meliputi jaringan 132.92.122.0/25 dan 132.92.121.0/24 dengan *amon* sebagai router perbatasan area. Kedua router perbatasan area tersebut terhubung langsung dengan backbone OSPF sehingga tidak perlu mendefinisikan *virtual link*.

Gambar 6.19 Jaringan OSPF dengan dua area

Konfigurasi routing yang diinginkan di jaringan ini adalah:

- Konfigurasi interface sama dengan konfigurasi tanpa area.
- Rute yang berasal dari routing statik dimasukkan ke OSPF sebagai ASE tipe 2 dengan metrik 50, kecuali rute 132.92.15.0/24 yang berasal dari *neith* diberi metrik 30.
- Area 1 mengeluarkan summary-LSA sesuai network-LSA dan tidak meringkasnya menjadi 132.92.36.0/24
- Area 2 dinyatakan sebagai stub area dan mengeluarkan summary-LSA dua jaringan yaitu 132.92.122.0/25 dan 132.92.121.0/24.

Konfigurasi routing di atas kemudian diturunkan menjadi konfigurasi GateD router OSPF. Router-router yang akan diperlihatkan konfigurasinya adalah *amon*, *neith*, *isis*, dan *khensu*. *Nephthys* adalah router backbone karena itu konfigurasi GateD-nya tetap seperti konfigurasi jaringan OSPF tanpa area. *Neith* dan *amon* adalah router perbatasan area (ABR) masing-masing untuk area 1 dan 2. Ketiga router tersebut merupakan router perbatasan autonomous system (ASBR). Dua router yang lain, *isis* dan *khensu* adalah router internal area, masing-masing tergabung dalam area 1 dan 2.

```
routerid 132.92.121.3 ;
rip no ;
ospf yes {
    backbone {
        authtype simple ;
        # jaringan point-to-point
        interface 132.92.128.66 cost 50 {
                priority 150 ;
                retransmitintreval 10 ;
                hellointerval 120 ;
                routerdeadinterval 1200 ;
                authkey "wanospf" ;
        } ;
        interface 132.92.128.129 cost 50 {
                priority 150 ;
                retransmitintreval 10 ;
                hellointerval 120 ;
                routerdeadinterval 1200 ;
                authkey "wanospf" ;
        } ;
    } ;
    area 2 {
        authtype simple ;
        stub cost 20 ;
        networks {
                132.92.121.0 masklen 24 ;
```

```
                    132.92.122.0 masklen 25 ;
            };
            interface 132.92.121.3 cost 10 {
                    priority 100 ;
                    hellointerval 10 ;
                    routerdeadinterval 40 ;
                    authkey "lanospf" ;
            };
        };
};
static {
    132.92.15 masklen 24 gateway 132.92.63.1 ;
    132.92.63.0 mask 255.255.255.0 gateway 132.92.63.1 ;
};
export proto ospfase type 1 {
    proto static {
        all metric 50 ;
    };
};
```

Gambar 6.20 Konfigurasi GateD di amon

```
routerid 132.92.36.122 ;
rip no ;
ospf yes {
    backbone {
        authtype simple ;
        # jaringan point-to-point
        interface 132.92.128.130 cost 50 {
                priority 250 ;
                retransmitintreval 10 ;
                hellointerval 120 ;
                routerdeadinterval 1200 ;
                authkey "wanospf" ;
        };
    };
    area 1 {
        authtype simple ;
        interface 132.92.36.122 cost 10 {
                priority 100 ;
```

```
                        hellointerval 10 ;
                        routerdeadinterval 40 ;
                        authkey "lanospf" ;
                };
                interface 132.92.36.171 cost 10 {
                        priority 250 ;
                        hellointerval 10 ;
                        routerdeadinterval 40 ;
                        authkey "lanospf" ;
                };
        };
};
static {
        171.21 masklen 16 gateway 171.21.13.8 ;
};
export proto ospfase type 1 {
        proto static {
                171.21 masklen 16 exact metric 50 ;
        };
};
```

Gambar 6.21 Konfigurasi GateD di neith

```
routerid 132.92.36.21 ;
rip no ;
ospf yes {
        area 1 {
                authtype simple ;
                interface 131.92.36.21 cost 10 {
                        priority 100 ;
                        hellointerval 10 ;
                        routerdeadinterval 40 ;
                        authkey "lanospf" ;
                };
                interface 131.92.36.36 cost 10 {
                        priority 100 ;
                        hellointerval 10 ;
                        routerdeadinterval 40 ;
                        authkey "lanospf" ;
                };
```

```
            interface 131.92.36.121 cost 10 {
                        priority 250 ;
                        hellointerval 10 ;
                        routerdeadinterval 40 ;
                        authkey "lanospf" ;
            };
      };
};
```

Gambar 6.22 Konfigurasi GateD di isis

```
routerid 132.92.122.3 ;
rip no ;
ospf yes {
      area 1 {
            authtype simple ;
            interface 131.92.122.3 cost 10 {
                        priority 100 ;
                        hellointerval 10 ;
                        routerdeadinterval 40 ;
                        authkey "lanospf" ;
            };
            interface 131.92.122.44 cost 10 {
                        priority 100 ;
                        hellointerval 10 ;
                        routerdeadinterval 40 ;
                        authkey "lanospf" ;
            };
      };
};
```

Gambar 6.23 Konfigurasi GateD di khensu

Dari konfigurasi GateD di *amon* dan *neith* terlihat bagaimana cara mendefinisikan sebuah router menjadi router perbatasan area. Untuk mendefinisikan router perbatasan area, pada GateD digunakan klausa **backbone** dan **area**. Klausa backbone meliputi semua interface yang terhubung ke backbone dan klausa area

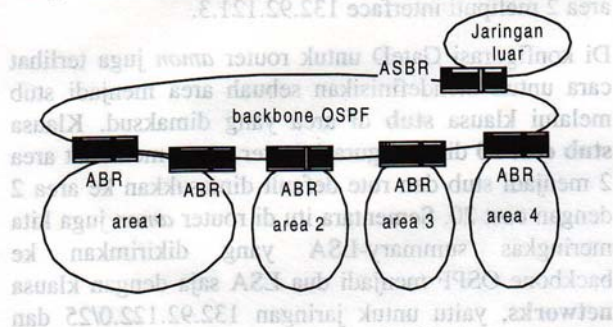
meliputi interface yang terhubung ke area yang bersangkutan. Router *amon* memiliki dua buah interface ke backbone dan sebuah interface ke area 2, sehingga klausa backbone meliputi inteface 132.92.128.66 dan 132.92.128.129, sedangkan klausa area 2 meliputi interface 132.92.121.3.

Di konfigurasi GateD untuk router *amon* juga terlihat cara untuk mendefinisikan sebuah area menjadi stub melalui klausa **stub** di area yang dimaksud. Klausa **stub cost 20** di konfigurasi router *amon* membuat area 2 menjadi stub dan rute default dimasukkan ke area 2 dengan cost 20. Sementara itu di router *amon* juga kita meringkas summary-LSA yang dikirimkan ke backbone OSPF menjadi dua LSA saja dengan klausa **networks**, yaitu untuk jaringan 132.92.122.0/25 dan 132.92.121.0/24.

Konfigurasi router internal area pada dasarnya tidak memiliki perbedaan dengan konfigurasi router pada jaringan OSPF tanpa area. Perbedaan hanya terletak pada klausa **area/backbone** untuk menentukan area tempat router-router tersebut berada.

6.6. Beberapa Tip dalam Merancang Jaringan OSPF

Kita telah mengetahui dasar-dasar dan karakteristik routing jaringan dengan OSPF dan sekarang yang perlu dilakukan adalah bagaimana merancang jaringan OSPF yang baik. Salah satu perhatian utama dalam perancangan routing adalah minimisasi tabel routing. Minimisasi ini dapat diperoleh dengan cara mengumpulkan alamat IP jaringan yang ber-dekatan di router-

router yang berdekatan pula. Dalam routing OSPF, minimisasi tabel routing dilakukan oleh router perbatasan area. Jadi, untuk merancang jaringan OSPF dengan baik, kita membagi jaringan OSPF tersebut menjadi beberapa area seperti pada Gambar 6.24.

Gambar 6.24 Konsep jaringan OSPF dengan area

Kita perlu merancang pengaturan pengalamatan IP pada setiap area sehingga setiap area terletak dalam satu jangkauan alamat IP jaringan, misal area 1 dialokasikan alamat IP 132.92.36.0/24, dan untuk area 2 diberi alamat IP 132.92.121.0/24. Dengan pengaturan alamat IP seperti ini kita dapat menyingkat summary-LSA yang dikirim ABR ke backbone OSPF menjadi satu jaringan dengan jangkauan IP yang besar.

Karakteristik OSPF yang lain adalah mendistribusikan AS-external-LSA ke seluruh area kecuali area stub. AS-external-LSA yang didistribusikan ke dalam area stub hanyalah untuk rute default sehingga kita dapat memperoleh tabel routing yang lebih pendek. Karena tabel routing di area stub lebih pendek, maka kita sebaiknya membuat seluruh area dalam jaringan OSPF tersebut menjadi area stub. Untuk memperoleh area

stub kita harus menjaga agar hanya router-router di backbone yang mengekspor rute dari protokol lain ke OSPF.

Routing link-state membutuhkan kemampuan CPU dan memori yang besar. Semakin banyak router yang menjalankan routing link-state, maka kemampuan CPU dan memori yang dibutuhkan juga semakin besar. Dalam OSPF, router-router dalam sebuah area menjalankan routing link-state yang sama dan setiap area menjalankan routing link-state yang berbeda. Dengan membagi jaringan OSPF menjadi beberapa area, router membutuhkan kemampuan CPU dan memori yang lebih sedikit. Untuk menjaga agar CPU dan memori yang diperlukan router tidak terlalu besar maka router di setiap area hendaknya dibatasi dalam jumlah puluhan.

OSPF juga mampu menghubungkan backbone yang terpisah dengan menggunakan jalur virtual (*virtual link*). Jalur virtual memiliki kelemahan karena jalur ini hanya dapat terbentuk setelah area transit konvergen. Jika area transit mengalami perubahan, backbone OSPF juga dapat berubah karena perubahan jalur virtual. Mengingat kelemahan ini, penggunaan jalur virtual dalam jaringan OSPF adalah suatu hal yang sangat perlu dihindari.

6.7. Ringkasan

Dalam bab ini telah dijelaskan implementasi konsep routing dalam jaringan TCP/IP menggunakan Unix FreeBSD. Protokol routing yang dibahas dalam bab ini adalah routing statik, RIP versi 2, dan OSPF.

Pembentukan routing di jaringan menggunakan routing statik terlihat memerlukan perhatian lebih banyak terutama untuk menjaga konsistensi routing. Jika jaringan hanya terdiri atas beberapa router, seperti pada contoh, maka pengaturan routing secara manual adalah pilihan yang wajar karena jumlah rute yang perlu dimasukkan hanya sedikit. Dari contoh tersebut dapat dibayangkan bagaimana repotnya jika kita ingin mengatur routing untuk jaringan yang besar secara manual.

Routing menggunakan RIP (versi 2) sangat sederhana dan ini terlihat dari konfigurasi GateD yang hanya terdiri atas satu baris untuk setiap interface jaringan. Perbandingan konfigurasi GateD RIPv2 dengan OSPF jelas sekali memperlihatkan bahwa OSPF lebih rumit daripada RIP. Konfigurasi GateD untuk protokol routing OSPF jauh lebih panjang daripada konfigurasi untuk RIP. Walaupun konfigurasi untuk OSPF lebih panjang, hal tersebut tentu lebih mudah dilakukan dalam jaringan yang besar daripada mengkonfigurasi routing secara manual.

Contoh di bab ini memperlihatkan bagaimana mengkonfigurasi jaringan OSPF dengan dan tanpa area. Pada prinsipnya kedua macam konfigurasi tersebut sama, kecuali pendefinisian area. Konfigurasi yang berbeda adalah di router perbatasan area (ABR) karena router ini mengurus lebih dari satu area. Pada ABR dilakukan minimisasi tabel routing melalui minimisasi summary-LSA dan pendefinisian area stub. Dari contoh tersebut juga diperlihatkan bagaimana mengekspor rute dari protokol routing yang lain ke OSPF.

OSPF merupakan protokol routing yang dapat digunakan pada jaringan TCP/IP yang besar. Dalam bab ini juga diberikan beberapa tip dalam merancang jaringan OSPF, yaitu:

- Bagi jaringan OSPF menjadi beberapa area
- Kumpulkan alamat IP jaringan yang dekat dalam satu area untuk memendekkan summary-LSA.
- Rancang area-area agar menjadi area stub untuk memperkecil tabel routing dalam area tersebut.
- Batasi jumlah router dalam setiap area.
- Jangan gunakan *virtual link* kecuali dalam keadaan yang tidak memungkinkan.

Bab 7
Simple Network Management Protocol

Ketika jumlah jaringan dalam sebuah organisasi semakin besar, dengan perangkat yang semakin banyak dan berbeda-beda dari berbagai vendor, maka saat itulah diperlukan sebuah kerangka manajeman jaringan yang koheren. Pada bagian ini, Anda akan mempelajari standard manajemen jaringan TCP/IP yang digunakan di internet.

Manajemen jaringan TCP/IP terdiri atas stasiun menajemen yang berkomunikasi dengan elemen-elemen jaringan. Elemen jaringan ini bisa berupa host, router, printer, dan sebagainya. Sedangkan stasiun manajemen biasanya berupa workstation dengan monitor berwarna dan grafis, yang menampilkan status elemen yang dimonitornya.

Untuk menajalankan aktifitas monitoring tersebut, antara manajer dan elemen-elemen jaringan yang dimonitor harus ada komunikasi. Ada dua arah komunikasi, pertama, manajer yang bertanya kepada elemen jaringan "berapa jumlah paket yang masuk ke interface et0?". Kedua, elemen jaringan yang

memberitahu manajer adanya kejadian penting seperti "interface et0 mati !". Selanjutnya stasiun manajemen akan menampilkan status interface tersebut di layarnya. Dengan cara seperti ini, seorang administrator jaringan dapat segera mengetahui adanya kegagalan dalam jaringannya.

Dalam jaringan TCP/IP, protokol aplikasi yang menangani soal manajemen jaringan ini adalah SNMP, singkatan dari *Simple Network Management Protocol*.

7.1. Apakah SNMP itu?

Secara sederhana, SNMP merupakan sebuah protokol yang didesain untuk memberikan kemampuan kepada pemakai untuk mengelola jaringan komputernya dari jarak jauh atau remote. Pengelolaan ini dilaksanakan dengan cara melakukan polling dan setting variabel-variabel elemen jaringan yang dikelolanya.

SNMP ini terdiri atas tiga elemen yaitu:

- MIB
- manajer
- agen

MIB atau *Management Information Base*, bisa dikatakan sebagai struktur database variabel elemen jaringan yang dikelola. Struktur ini bersifat hierarki dan memiliki aturan sedemikian rupa sehingga informasi nilai setiap variabel dapat diketahui atau diset dengan mudah.

Agen merupakan software yang dijalankan disetiap node atau elemen jaringan yang akan dimonitor.

Tugasnya adalah mengumpulkan seluruh informasi yang telah ditentukan dalam MIB.

Manajer merupakan software yang berjalan di sebuah host di jaringan. Manajer ini bertugas mengumpulkan informasi dari agen-agen. Tidak semua informasi yang dimiliki oleh agen diminta oleh manajer. Informasi-informasi yang diminta oleh administrator jaringan, yang menjalankan host yang berfungsi sebagai manajer, saja yang akan dikumpulkan dari agen.

7.2. Protokol SNMP

SNMP didesain oleh IETF untuk pemakaian di Internet. Saat ini SNMP didesain di atas protokol UDP (*User Datagram Protocol*) seperti pada Gambar 7.1. Karena menggunakan protokol UDP, SNMP adalah protokol yang *connectionless*. Tidak ada jaminan lalu lintas manajemen diterima oleh entitas lain dengan sempurna. Dengan protokol ini, overhead proses dapat dikurangi dan diperoleh kesederhanaan. Jika reliabilitas dan accountabilitas yang diperlukan, manajer jaringan harus membangun operasi yang *connection oriented* pada aplikasi di lapisan atasnya.

SNMP ini bekerja secara sederhana. Manajer dan agen saling berkirim pesan berupa permintaan manajer dan jawaban dari agen tentang informasi jaringan. Pesan-pesan ini dibawa oleh paket-paket data yang disebut PDU, *Protocol Data Unit*.

Aplikasi Manajemen Jaringan
Agent
SNMP
UDP
IP
Lapisan Bawah

Gambar 7.1 Lapisan-lapisan SNMP

7.3. PDU SNMP

PDU atau *Protocol Data Unit* merupakan unit data yang terdiri atas sebuah *header* dan beberapa data yang ditempelkan. Dilihat dari perspektif di atas, PDU ini dapat dilihat sebagai sebuah benda yang mengandung variabel-variabel. Variabel ini memiliki nama dan nilai.

Protokol SNMP menggunakan operasi yang relatif sederhana dan PDU dalam jumlah terbatas untuk menjalankan fungsinya. Lima PDU yang telah didefinisikan dalam standard adalah sebagai berikut:

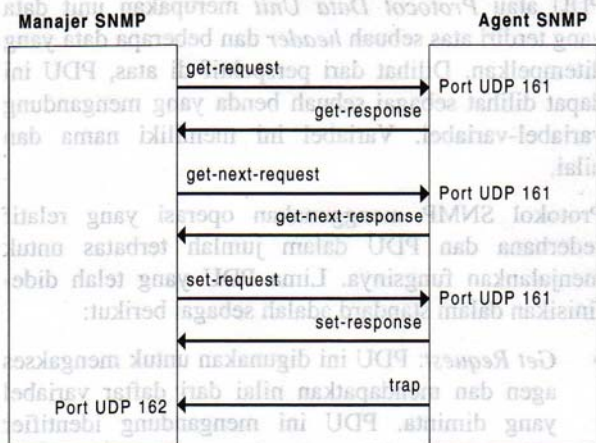
- *Get Request*: PDU ini digunakan untuk mengakses agen dan mendapatkan nilai dari daftar variabel yang diminta. PDU ini mengandung identifier yang membedakannya dengan *multi request* ataupun nilai variabel (status elemen jaringan).

- *Get-Next Request*: Seperti *Get Request*, tetapi memungkinkan pengambilan informasi pada logical identifier selanjutnya dalam MIB Tree secara berurutan.

- *Get Response*: PDU ini untuk merespon unit data *Get Request, Get-Next Request* dan *Set Request*, jadi dikeluarkan oleh agent

- *Set Request*: Dipakai untuk menjelaskan aksi yang harus dilaksanakan di elemen jaringan. Biasanya untuk mengubah nilai suatu daftar variabel.

- *Trap*: PDU ini memungkinkan modul manajemen jaringan (agent) memberi laporan tentang kejadian pada elemen jaringan kepada manajer.

Gambar 7.2 menggambarkan pesan yang dipertukarkan di antara manajer dan agent.

Gambar 7.2 Lima operator SNMP

Karena keempat PDU merupakan protokol *request-reply* yang sederhana, maka SNMP menggunakan UDP. Hal ini berarti bahwa *request* dari manajer bisa jadi tidak sampai ke agen dan *reply* dari agen bisa tidak sampai ke manajer. Untuk itu, manajer menerapkan suatu nilai *timeout* dan *retransmisi.*

Tiga jenis *request* dari manajer dikirim ke port UDP 161. Dan agen mengirim *trap* ke port UDP 162. Dengan menggunakan dua port yang berbeda, sebuah host bisa menjalankan fungsi sebagai manajer dan agent sekaligus.

7.4. Struktur Informasi dalam SNMP

Infomasi dalam SNMP disimpan dalam bentuk variabel-variabel yang didefinisikan dalam MIB. Ada variabel berjenis teks, bilangan bulat atau integer, waktu, dan sebagainya. Jenis-jenis ini juga telah didefinisikan ke dalam beberapa tipe data atau variabel.

SNMP menggunakan beberapa tipe data, antara lain:

INTEGER. Ada tiga cara pendefinisian variabel:

Ditulis apa adanya tanpa ketentuan batasan nilai. Misal MTU sebuah interface.

Dengan mengambil nilai-nilai tertentu. Misal jika 1, maka IP forwarding diaktifkan, dan 2 jika tidak diaktifkan.

Dengan mendefinisikan nilai maksimum dan minimumnya. Misal jumlah port UDP dan TCP, dari 0 sampai 65535.

OCTET STRING. String atau teks yang dinyatakan oleh byte-byte 8-bit, yang mempunyai nilai dari 0 sampai 255.

DisplayString. String yang dinyatakan oleh byte-byte 8-bit, dan harus merupakan karakter dari NVT ASCII.

OBJECT IDENTIFIER. Tipe data yang menyatakan objek tertentu (yang telah ditetapkan oleh suatu organisasi, tidak secara random). Akan dibahas selanjuntya.

NULL. Variabel yang bersangkutan tidak punya nilai. Misal, jika nilai suatu variabel tidak diset, maka operasi get atau get-next untuk variabel tersebut akan memberikan hasil NULL.

IpAddress. Sebuah OCTET STRING dengan panjang 4 byte untuk IP Address.

PhysAddress. Sebuah OCTET STRING yang menyatakan sebuah alamat fisik, misal alamat Ethernet 6-byte.

Counter. Integer non-negatif yang nilainya naik secara monoton dari 0 sampai $2^{32} - 1$ (4.294.967.295) dan kemudian kembali lagi ke 0.

Gauge. Integer non-negatif antara 0 dan $2^{32} - 1$ yang nilainya bisa naik atau turun, dan akan tetap pada nilai maksimumnya.

TimeTicks. Sebuah counter yang menghitung waktu setiap detik seratus kali. Misalnya variabel ssyUpTime adalah jumlah seper-seratus detik agent telah dihidupkan.

SEQUENCE. Sama dengan sebuah struktur dalam bahasa pemrograman C. Contohnya, MIB mendefinisi-

kan sebuah `SEQUENCE` (struktur) bernama `UdpEntry` yang berisi informasi tentang port UDP yang sedang aktif atau sedang digunakan oleh aplikasi. Ada dua entri dalam struktur tersebut, yaitu:

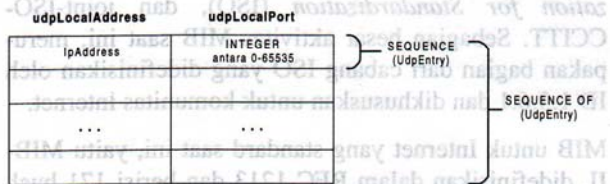
udpLocalAddress, yang memiliki tipe IpAddress untuk menyatakan IP address lokal.

udpLocalPort, dengan tipe INTEGER untuk menyatakan nomor port lokal

SEQUENCE OF. Merupakan definisi sebuah vektor (array satu dimensi), dengan semua elemen memiliki tipe data yang sama. SNMP juga menggunakan tipe data ini dengan setiap elemen vektor berupa `SEQUENCE` (struktur). Kita anggap ini sebagai aray dua dimensi atau tabel.

Misalnya, tabel UDP bernama `udpTable` dan merupakan `SEQUENCE OF` 2-elemen `SEQUENCE` (struktur) `UdpEntry` seperti yang telah didefinisikan di atas (di bagian `SEQUENCE`). Array dua dimensi ini dapat dilihat pada Gambar 7.3 berikut:

udpLocalAddress	udpLocalPort
IpAddress	INTEGER antara 0-65535
...	...

SEQUENCE (UdpEntry)
SEQUENCE OF (UdpEntry)

Gambar 7.3 Tabel UDP listener (`udpTable`) sebagai aray dua dimensi dalam SNMP

Jumlah baris dalam tabel array tidak didefinisikan oleh SNMP, tetapi dengan operator *get-next* yang digunakan terus-menerus dapat diketahui dimana akhir barisnya.

7.5. Sekilas MIB dan Object Identifier

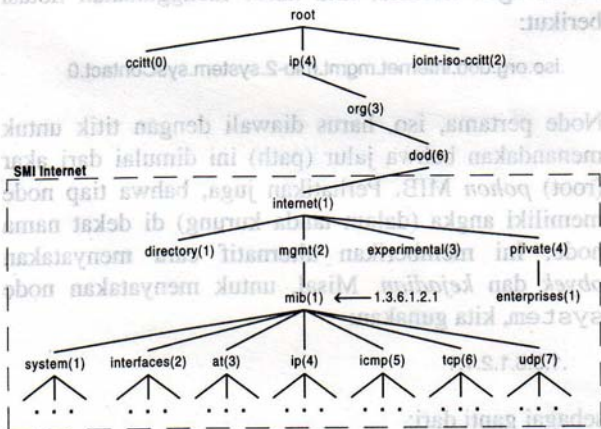
MIB, *Management Information Base*, dapat digambarkan sebagai sebuah *pohon* abstrak yang memiliki sebuah akar. Akar ini tidak punya nama. Item-item data secara individual membentuk *daun-daun*nya. *Object identifier* atau ID, mengidentifikasi atau memberi nama objek-objek dalam *pohon* MIB. Penamaan ini dilakukan secara unik.

ID dari objek-objek tersebut mirip dengan nomor telepon, yang diorganisasikan secara hierarki. Masing-masing objek memiliki angka tertentu yang menunjukkan organisasi tertentu sebagai pemiliknya.

Struktur ID objek MIB mendefinisikan tiga cabang utama: *Consultative Committee for International Telegraph and Telephone* (CCITT), *International Organization for Standardization* (ISO), dan joint-ISO-CCITT. Sebagian besar aktivitas MIB saat ini, merupakan bagian dari cabang ISO yang didefinisikan oleh ID 1.3.6.1 dan dikhususkan untuk komunitas internet.

MIB untuk Internet yang standard saat ini, yaitu MIB-II, didefinisikan dalam RFC 1213 dan berisi 171 buah objek. Objek-objek ini dikelompokkan berdasarkan protokol serta beberapa kategori lainnya, termasuk `system`, dan *interfaces*.

Pohon MIB ini dapat berkembang, karena memiliki cabang *experimental* dan *private*. Para vendor dapat mendefinisikan cabang-cabang private mereka sendiri untuk mengakomodasi objek kejadian (*instances*) dari produk-produk mereka.

Struktur dasar *pohon* MIB diperlihatkan oleh Gambar 7.4 di bawah ini.

Gambar 7.4 Object identifier dalam MIB

Sistem tersebut memiliki beberapa node anak seperti sysDescr, sysContact, sysName, dan sysLocation. Node anak ini disebut obyek, dan dalam *pohon* MIB disebut *daun*. Sebenarnya menyebutnya sebagai *daun* itu kurang tepat karena *obyek* tidak memiliki nilai, namun yang memiliki nilai adalah *obyek kejadian* (instance). Obyek kejadian inilah yang menyimpan nilai sebuah node *daun*.

Perbedaan antara *obyek* dan *kejadian* menjadi semakin jelas ketika kita memikirkan cara menyatakan informasi MIB. Untuk menyatakan obyek `sysContact`, kita gunakan notasi:

.iso.org.dod.internet.mgmt.mib-2.system.sysContact

Tetapi untuk menyatakan nilai `sysContact` dari mesin agen tertentu, kita harus menggunakan notasi berikut:

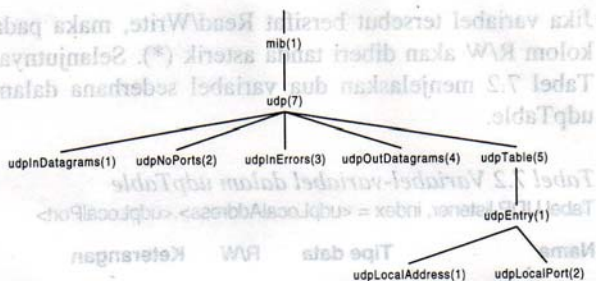
.iso.org.dod.internet.mgmt.mib-2.system.sysContact.0

Node pertama, iso, harus diawali dengan titik untuk menandakan bahwa jalur (path) ini dimulai dari akar (root) *pohon* MIB. Perhatikan juga, bahwa tiap node memiliki angka (dalam tanda kurung) di dekat nama node. Ini memberikan alternatif cara menyatakan *obyek* dan *kejadian*. Misal, untuk menyatakan node `system`, kita gunakan:

.1.3.6.1.2.1.1

sebagai ganti dari:

.iso.org.dod.internet.mgmt.mib-2.system

Kita kembali melihat gambar *pohon* MIB di atas. MIB dibagi-bagi kedalam beberapa group yaitu `system`, `interfaces`, `at` (address translation), `ip`, dan seterusnya. Kita akan melihat variabel-variabel dalam group UDP. Ini merupakan group yang sederhana yang terdiri atas beberapa variabel dan sebuah tabel. Gambar 7.5 memperlihatkan struktur group UDP.

Gambar 7.5: Struktur group UDP

Ada empat variabel sederhana dan sebuah tabel yang berisi dua variabel sederhana. Tabel berikut menjelaskan keempat variabel sederhana itu.

Tabel 7.1 Variabel-variabel sederhana dalam group UDP

Nama	Tipe data	R/W	Keterangan
UdpInDatagrams	Counter		Jumlah datagram UDP dikirim ke proses pemakai
UdpNoPorts	Counter		Jumlah datagram UDP diterima untuk proses aplikasi yang tidak sampai di port tujuan
UdpInErrors	Counter		Jumlah datagram UDP tidak terkirim karena alasan selain udpNoPorts.
UdpOutDatagrams	Counter		Jumlah datagram UDP yang dikirim

Jika variabel tersebut bersifat Read/Write, maka pada kolom R/W akan diberi tanda asterik (*). Selanjutnya, Tabel 7.2 menjelaskan dua variabel sederhana dalam udpTable.

Tabel 7.2 Variabel-variabel dalam udpTable

Tabel UDP listener, index = <udpLocalAddress>.<udpLocalPort>

Nama	Tipe data	R/W	Keterangan
udpLocalAddress	IpAddress		IP address lokal bagi UPD listener ini. 0.0.0.0 artinya listener akan menerima datagram dari semua interface.
udpLocalPort	[0...65535]		Nomor port lokal dari listener ini.

Baris pertama dari tabel di atas menyatakan nilai dari index yang akan digunakan sebagai referensi setiap baris tabel.

7.6. Identifikasi Kejadian (instance)

Setiap variabel dalam MIB harus diidentifikasi ketika SNMP akan merujuk kepadanya, mengambil atau mengeset nilainya. Hanya node *daun* yang akan dirujuk. SNMP tidak akan memanipulasi semua baris atau kolom dalam tabel. Kembali ke Gambar 7.5, yang disebut node *daun* adalah empat variabel seperti diuraikan dalam Tabel 7.1 dan dua variabel dalam Tabel 7.2. Variabel-variabel seperti mib, udp, udpTable dan udpEntry bukan node *daun*. Sekarang semakin jelas mana yang disebut sebagai node *daun* dan yang bukan.

7.6.1. Variabel Sederhana

Variabel sederhana dirujuk dengan menambahkan ".0" kepada object identifier variabel. Misalnya untuk counter udpInDatagrams dari Tabel 7.1, dengan object identifier 1.3.6.1.2.1.7.1, dirujuk dengan cara menuliskannya sebagai 1.3.6.1.2.1.7.1.0. Nama secara tekstual dari rujukan ini adalah iso.org.dod. internet.mgmt.mib.udp.udpInDatagrams.0.

7.6.2. Tabel

Identifikasi kejadian dari entri tabel itu lebih detail lagi. Kembali ke Gambar 7.5 untuk UDP listener. Untuk tabel UDP listener, MIB mendefinisikan index sebagai kombinasi dari dua variabel yaitu udpLocalAddress (berupa IP address) dan udpLocalPort (berupa integer) yang diperlihatkan pada baris paling atas di Tabel 7.2.

Anggap ada tiga baris dalam tabel UDP listener: baris pertama adalah IP address 0.0.0.0 dan port 67, kedua untuk 0.0.0.0 dan port 161, dan ketiga untuk 0.0.0.0 dan port 520. Lihat Tabel 7.3.

Tabel 7.3 Contoh tabel UDP listener

udpLocalAddress	udpLocalPort
0.0.0.0	67
0.0.0.0	161
0.0.0.0	520

Tabel tersebut menunjukkan bahwa sistem akan menerima datagram UDP dari semua interface untuk port 67 (server BOOTP), 161 (SNMP, dan 520 (RIP). Ketiga baris ini dinyatakan seperti pada Tabel 7.4.

Tabel 7.4 Identifikasi kejadian untuk baris-baris dalam tabel UDP listener

Baris	Object Identifier	Nama yang disingkat	Nilai
1	1.3.6.1.2.1.7.5.1.0.0.0.0.67	udpLocalAddress.0.0.0.0.67	0.0.0.0
	1.3.6.1.2.1.7.5.2.0.0.0.0.67	udpLocalPort.0.0.0.0.67	67
2	1.3.6.1.2.1.7.5.1.0.0.0.0.161	udpLocalAddress.0.0.0.0.161	0.0.0.0
	1.3.6.1.2.1.7.5.2.0.0.0.0.161	udpLocalPort.0.0.0.0.161	161
3	1.3.6.1.2.1.7.5.1.0.0.0.0.520	udpLocalAddress.0.0.0.0.520	0.0.0.0
	1.3.6.1.2.1.7.5.2.0.0.0.0.520	udpLocalPort.0.0.0.0.520	520

7.6.3. Urutan Lexicographic

Kalau kita mengakses variabel MIB dengan `snmpwalk`, maka akan diperoleh hasil yang diurutkan berdasarkan urutan object identifiernya. Artinya, nilai-nilai variabel dalam satu kolom akan ditampilkan, baru dilanjutkan dengan variabel pada kolom selanjutnya. Urutan lexicographic ini ditunjukkan oleh Tabel 7.5 berikut.

Tabel 7.5 Urutan lexicographic, atau urutan kolom-baris.

udpLocalAddress		udpLocalPort	
0.0.0.0		67	
0.0.0.0		161	
0.0.0.0		520	

7.7. Contoh Sederhana

Kita akan melihat beberapa contoh cara bekerja dengan SNMP. Software yang digunakan di sini adalah buatan Carnegie Melon University (CMU) yang berjalan di atas sistem operasi LINUX. Versi yang dipakai adalah versi 3.2. Distribusi source-code dan biner dapat diambil di **ftp.ibr.cs.tu-bs.de** di direktori

/pub/local/linux-cmu-snmp. File source-code adalah cmu-snmp-linux-3.2-src.tar.gz, sedangkan file binernya adalah cmu-snmp-linux-3.2-bin.tar.gz.

7.7.1. Instalasi Agen

Contoh yang digunakan dalam buku ini telah dilakukan pada mesin Linux Slakware, dengan kernel versi 2.0.29. Langkah pertama instalasi adalah mengambil software dari **ftp.ibr.cs.tu-bs.de** untuk source-code. Software ini adalah software SNMP versi 2 untuk agent dan manajer.

Selanjutnya dilakukan kompilasi program di mesin Linux Kopi file **cmu-snmp-linux-3.2-src.tar.gz** ke direktori **/usr/local.**

Uraikan file yang terkompresi tersebut, dengan perintah:

```
gunzip cmu-snmp-linux-3.2-src.tar.gz dan
tar –xvf cmu-snmp-linux-3.2-src.tar
```

Selanjutnya akan diperoleh direktori baru yang berisi file-file yang diperlukan untuk kompilasi program, yaitu direktori cmu-snmp-linux-3.2/

Masuk ke direktori /usr/local/cmu-snmp-linux-3.2/

Konfigurasikan sistem sebelum dikompilasi dengan perintah berikut:

```
./configure
```

Olah (compile) source-code agar diperoleh program yang siap dijalankan, dengan printah berikut:

```
make
```

Install program yang telah diolah agar dicopy ke direktori-direktori yang seharusnya, dengan perintah berikut:

```
make install
```

Edit file /etc/snmpd.conf sesuai dengan sistem lokal, khususnya untuk entri di bawah ini:

```
community nipassword   xmini  -       public
community iniprivate   mini   mini    private
system contact:     Ismail Fahmi
system location:    CNRG ITB
system name:        Khensu.cnrg.net
```

Anda dapat mengganti entri di atas sesuai dengan sistem yang Anda gunakan. Community name secara default diset **public** dan **private**. Jika Anda ingin agar agen hanya dapat diakses oleh Anda sendiri, gunakan *community name* yang unik, seperti **inipassword** untuk akses oleh umum dan iniprivate untuk akses khusus (misal untuk mengeset sebuah nilai, lihat perintah **snmpset**).

Tambahkan dalam file **/etc/rc.d/rc.local** baris berikut untuk menjalankan agen snmpd setiap kali sistem dihidupkan:

```
/usr/sbin/snmpd -f ; echo 'starting snmpd'
```

Atau jalankan secara manual agen snmpd dengan mengetik

```
snmpd –f
```

Pilihan –f artinya program diset agar berjalan sebagai background.

Ujilah program yang telah dijalankan dengan perintah berikut:

```
snmpwalk localhost inipassword   system
```

Jika instalasi program sudah benar, akan diperoleh respon seperti di bawah ini:

```
khensu:/etc# snmpwalk localhost inipassword system
system.sysDescr.0 = "Linux version 2.0.29 (ismail@khensu) (gcc
version 2.7.2.1) #6 Fri Jul 18 01:24:33 JVT 1997"
system.sysObjectID.0 = OID: enterprises.tubs.ibr.linuxMIB
system.sysUpTime.0 = Timeticks: (1970) 0:00:19
system.sysContact.0 = "Ismail Fahmi"
system.sysName.0 = "Khensu.cnrg.net"
system.sysLocation.0 = "CNRG ITB"
system.sysServices.0 = 72
system.sysORLastChange.0 = Timeticks: (1972) 0:00:19
system.sysORTable.sysOREntry.sysORID.1 = OID:
enterprises.tubs.ibr.linuxMIB.linuxAgents.1
system.sysORTable.sysOREntry.sysORDescr.1 = "LINUX
agent"
system.sysORTable.sysOREntry.sysORUpTime.1 = Timeticks:
(1974) 0:00:19
```

Jika Anda mengakses dengan nama community yang salah, akan keluar respon seperti ini:

```
khensu:/etc# snmpwalk localhost public system
An error occurred, Quitting
```

7.7.2. Perintah-perintah untuk mengakses agen

Selain agen, software ini juga menyediakan perintah-perintah yang dapat digunakan untuk mengakses agen tersebut. Perintah-perintah dasar yang akan digunakan dalam buku ini antara lain: snmpget, snmpgetnext, snmpwalk, dan snmpset.

snmpget

Perintah `snmpget` digunakan untuk mengambil nilai sebuah variabel MIB dari sebuah agen. Kita harus tahu identifikasi kejadian (node *daun*) secara tepat. Aturan penulisan perintahnya adalah:

```
snmpget hostname community_name MIB_object_instance
```

Contoh:

```
khensu:~$ snmpget localhost inipassword
system.sysContact.0
system.sysContact.0 = "Ismail Fahmi"
```

snmpgetnext

Perintah `snmpgetnext` digunakan untuk mengambil nilai sebuah obyek kejadian setelah obyek kejadian yang disebutkan dalam perintah (**MIB_object_instance**). Aturan penulisan perintahnya adalah:

```
snmpgetnext hostname community_name
MIB_object_instance
```

Hasilnya adalah nilai obyek kejadian yang urutannya setelah obyek kejadian yang disebutkan dalam **MIB_object_instance** tersebut.

Contoh:

```
khensu:~$ snmpgetnext localhost inipassword
system.sysContact.0
system.sysName.0 = "Khensu.cnrg.net"
```

Setelah `sysContact.0`, obyek kejadian berikutnya adalah `sysName.0`. Pada contoh di atas, `sysName.0` berisi nama mesin yang ditempati oleh agen.

snmpwalk

Perintah `snmpwalk` digunakan untuk mengambil nilai satu atau lebih variabel MIB dari agen tanpa kita harus menyatakan identitas kejadiannya secara tepat. Aturan penulisan perintahnya adalah:

```
snmpwalk hostname community_name MIB_object_type
```

Contoh:

```
khensu:~$ snmpwalk localhost inipassword
system.sysContact
system.sysContact.0 = "Ismail Fahmi"
```

snmpset

Perintah `snmpset` digunakan untuk mengeset nilai sebuah variabel MIB. Aturan penulisan perintahnya adalah:

```
snmpset hostname community_name MIB_object_instance type
value
```

Tipe (type) pada perintah di atas adalah tipe data dari **MIB_object_instance**. Aturannya adalah:

```
i: INTEGER, s: STRING, x: HEX STRING, d: DECIMAL STRING
    n: NULLOBJ, o: OBJID,  t: TIMETICKS,  a: IPADDRESS
```

Contoh, kita akan mengganti nama `sysContact` dengan alamat emailnya. Tipe datanya adalah `STRING` atau `s`.

```
khensu:~$ snmpset localhost inipassword system.sysContact.0 s
ismail@khnemu.cnrg.net
Error in packet.
Reason: There is no such variable name in this MIB.
This name doesn't exist: system.sysContact.0
```

Jika kita menggunakan inipassword sebagai community_namenya, akan diperoleh pesan kesalahan seperti di atas. Ini disebabkan untuk mengeset nilai variabel MIB harus menggunakan community_name untuk private. Community_name private untuk sistem dalam buku ini diset sebagai iniprivate. Selanjutnya kita ulangi perintah di atas dengan mengganti community_namenya.

```
khensu:~$ snmpset localhost iniprivate system.sysContact.0 s
ismail@khnemu.cnrg.net
system.sysContact.0 = "ismail@khnemu.cnrg.net"
system.sysContact.0 = "ismail@khnemu.cnrg.net"
```

dan berhasil.

7.8. Lebih Jauh tentang MIB

Sekarang kita kembali ke MIB dan selanjutnya akan kita kupas penjelasan bagi masing-masing group, yang akan disertai dengan contoh. Group yang akan dijelaskan di sini hanya meliputi group: system, if, at, ip, icmp, dan tcp.

7.8.1. Group system

Group `system` ini sederhana, terdiri atas tujuh variabel yang sederhana (tanpa tabel). Tabel 7.6. memperlihatkan nama, tipe data dan penjelasannya.

Kita dapat mengakses informasi variabel-variabel di atas dari agen, dengan perintah sebagai berikut:

```
khensu:~$ snmpget localhost inipassword system.sysDescr.0
      system.sysObjectID.0 system.sysUpTime.0
system.sysServices.0
system.sysDescr.0 = "Linux version 2.0.29 (ismail@khensu) (gcc version
2.7.2.1) #6 Fri Jul 18 01:24:33 JVT 1997"
system.sysObjectID.0 = OID: enterprises.tubs.ibr.linuxMIB
system.sysUpTime.0 = Timeticks: (6154624) 17:05:46
system.sysServices.0 = 72
```

Object identifier sistem ini adalah dalam group internet.private.enterprises.tubs.ibr.linuxMIB. Kita bisa lihat, variabel sysServices 72 merupakan jumlah dari 64 (aplikasi) dan 8 (transport).

Tabel 7.6 Variabel-variabel dalam group system

Nama	Tipe data	R/W	Keterangan
sysDescr	DisplayString		Penjelasan tekstual dari entitas
sysObjectID	ObjectID		ID vendor dalam sub-tree 1.3.6.1.4.1
sysUpTime	TimeTicks		Waktu dalam seper seratus detik sejak agen manajemen jaringan dijalankan.
sysContact	DisplayString	*	Nama penanggung jawab dan cara menghubunginya
sysName	DisplayString	*	Nama domain dari node yang ditulis secara FQDN
sysLocation	DisplayString	*	Lokasi fisik dari node

Nama	Tipe data	R/W	Keterangan
sysServices	[0...127]		Nilai yang menandakan layanan yang disediakan oleh node. Nilai ini merupakan jumlah dari layer-layer model OSI yang didukung oleh node. Nilai-nilai yang dijumlah-kan bergantung pada layanan yang disedia-kan, yang bisa meliputi: 0x01 (1=fisik), 0x02 (2=datalink), 0x04 (4=internet), 0x08 (8=transport/end-to-end), dan 0x40 (64=aplikasi).

7.8.2. Group interfaces

Group ini hanya memiliki satu variabel yaitu jumlah interface dalam sistem, seperti ditunjukkan oleh tabel berikut.

Tabel 7.7 Variabel sederhana dalam group if

Nama	Tipe data	R/W	Keterangan
ifNumber	INTEGER		Jumlah interface jaringan dalam sistem

Group ini juga mendefinisikan sebuah tabel dengan 22 buah kolom. Setiap baris tabel mendefinisikan karakteristik setiap interface, seperti terlihat pada tabel 7.8.

Tabel 7.8 Variabel dalam tabel interface: ifTable

Tabel interface, index = <IfIndex>			
Nama	Tipe data	R/W	Keterangan
ifIndex	INTEGER		Nomor index dari interface, antara satu dan

Tabel interface, index = <IfIndex>			
Nama	Tipe data	R/W	Keterangan
			ifNumber
ifDescr	Display String		Deskripsi tekstual tentang interface
ifType	INTEGER		Tipe, misalnya: 6 = Ethernet, 7 = 802.3 Ethernet, 9 = 802.5 token ring, 23 = PPP, 28 = SLIP
ifMtu	INTEGER		MTU (maximum transmission unit) dari interface
ifSpeed	Gauge		Kecepatan dalam bit per detik
ifPhysAddress	PhysAddress		Alamat fisik, atau string dengan panjang 0 untuk interface tanpa alamat fisik (misalnya link serial)
ifAdminStatus	[1...3]	*	Status interface yang diharapkan: 1=up, 2=down, 3=testing.
ifOperStatus	[1...3]		Status interface saat ini: 1=up, 2=down, 3=testing.
ifLastChange	TimeTicks		Nilai sysUpTime ketika interface memasuki kondisi operasional saat ini.
ifInOctets	Counter		Jumlah total byte yang diterima, termasuk karakter frame
ifInUcastPkts	Counter		Jumlah paket unicast yang dikirim ke layer lebih tinggi.
ifInNUcastPkts	Counter		Jumlah paket non-unicast (misalnya broad-cast atau

Tabel interface, index = <IfIndex>			
Nama	Tipe data	R/W	Keterangan
			multicast) yang dikirim ke layer lebih tinggi.
ifInDiscards	Counter		Jumlah paket diterima yang dibuang (discarded) meski bukan paket rusak (misalnya karena buffer penuh).
ifInErrors	Counter		Jumlah paket diterima yang dibuang karena rusak.
ifInUnknownProtos	Counter		Jumlah paket diterima yang dibuang karena protokol tidak dikenal.
ifOutOctets	Counter		Jumlah total byte yang diterima, termasuk karakter frame
ifOutUcastPkts	Counter		Jumlah paket unicast yang diterima dari layer lebih tinggi.
ifOutNUcastPkts	Counter		Jumlah paket nonunicast (misalnya broadcast atau multicast) yang diterima dari layer lebih tinggi.
ifOutDiscards	Counter		Jumlah paket dikirim yang dibuang (discarded) meski bukan paket rusak (misalnya karena buffer penuh).
ifOutErrors	Counter		Jumlah paket dikirim yang dibuang karena rusak.
ifOutQLen	Gauge		Jumlah paket dalam antrian keluar.
ifSpesific	ObjectID		Referensi ke definisi MIB khusus buat tipe media tertentu ini.

Kita lihat beberapa jumlah interface yang dimiliki mesin Khensu:

```
khensu:~$ snmpget localhost inipassword
interfaces.ifNumber.0
interfaces.ifNumber.0 = 10
```

Ada sepuluh interface, baik bersifat fisik maupun *pseudo*, yang dimiliki oleh mesin Linux. Kita lihat kejadian selanjutnya setelah **interfaces.ifNumber.0** dengan snmpgetnext:

```
khensu:~$ snmpgetnext localhost inipassword
interfaces.ifNumber.0
interfaces.ifTable.ifEntry.ifIndex.1 = 1
```

Sekarang kita akan lihat beberapa variabel dalam tabel di atas dengan perintah `snmpget`. Kita ambil contoh untuk interface dengan ifIndex 1dan 4.

```
khensu:~$ snmpget localhost inipassword
interfaces.ifTable.ifEntry.ifDescr.1
interfaces.ifTable.ifEntry.ifType.1
interfaces.ifTable.ifEntry.ifMtu.1
interfaces.ifTable.ifEntry.ifSpeed.1
interfaces.ifTable.ifEntry.ifPhysAddress.1
interfaces.ifTable.ifEntry.ifDescr.1 = "lo0" Hex: 6C 6F 30
interfaces.ifTable.ifEntry.ifType.1 = 24
interfaces.ifTable.ifEntry.ifMtu.1 = 3584
interfaces.ifTable.ifEntry.ifSpeed.1 = Gauge: 20000000
interfaces.ifTable.ifEntry.ifPhysAddress.1 =  Hex: 00 00 00 00 00 00
khensu:~$ snmpget localhost inipassword
interfaces.ifTable.ifEntry.ifDescr.4
interfaces.ifTable.ifEntry.ifType.4 interfaces.ifTable.ifEntry.ifMtu.4
interfaces.ifTable.ifEntry.ifSpeed.4
interfaces.ifTable.ifEntry.ifPhysAddress.4
interfaces.ifTable.ifEntry.ifDescr.4 = "ed0" Hex: 65 74 68 30
interfaces.ifTable.ifEntry.ifType.4 = ethernet-csmacd(6)
interfaces.ifTable.ifEntry.ifMtu.4 = 1500
interfaces.ifTable.ifEntry.ifSpeed.4 = Gauge: 10000000
interfaces.ifTable.ifEntry.ifPhysAddress.4 =  Hex: 00 80 48 A8 C0 90
```

Interface dengan index 1 adalah interface loopback (tipe 24) sedangkan ethernet card (tipe 6) merupakan interface dengan index 4.

Obyek kejadian ifPhysAddress.4 merupakan alamat fisik dari interface. Kita bisa cek menggunakan perintah ifconfig eth0:

```
khensu:~$ ifconfig eth0
eth0    Link encap:10Mbps Ethernet HWaddr 00:80:48:A8:C0:90
        inet addr:132.92.121.114 Bcast:132.92.121.127 Mask:255.255.255.224
        UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
           RX packets:64443 errors:0 dropped:0 overruns:0
           TX packets:70650 errors:0 dropped:0 overruns:0
           Interrupt:5 Base address:0x280
```

Diperoleh alamat fisik yang sama untuk ethernet (eth0) tersebut yaitu: 00:80:48:A8:C0:90.

7.8.3. Group at (*address translation*)

Hanya ada sebuah tabel dengan tiga kolom yang didefinisikan untuk group at seperti ditunjukkan oleh tabel berikut.

Tabel 7.9 Tabel Address Translation

Tabel Address Translation, index = *<atIfIndex>.1.<atNetAddress>*

Nama	Tipe data	R/W	Keterangan
atIfIndex	INTEGER	*	Nomor interface: ifIndex
atPhysAddress	PhysAddress	*	Alamat fisik.
atNetAddress	NetworkAddress	*	IP address

Kita gunakan **snmpwalk** untuk mengakses variabel-variabel di atas untuk mesin khnemu, dan hasilnya adalah urutan lexicographic dari obyek kejadian di dalam group at.

```
khensu:~$ snmpwalk khnemu public at
at.atTable.atEntry.atIfIndex.1.1.132.92.122.17 = 1
at.atTable.atEntry.atIfIndex.1.1.132.92.122.18 = 1
at.atTable.atEntry.atIfIndex.1.1.132.92.122.19 = 1
at.atTable.atEntry.atIfIndex.1.1.132.92.122.30 = 1
at.atTable.atEntry.atPhysAddress.1.1.132.92.122.17 = Hex: 00 80 C8 3F 9E 15
at.atTable.atEntry.atPhysAddress.1.1.132.92.122.18 = Hex: 00 00 0C 75 B0 AF
at.atTable.atEntry.atPhysAddress.1.1.132.92.122.19 = Hex: 00 80 48 E4 11 06
at.atTable.atEntry.atPhysAddress.1.1.132.92.122.30 = Hex: 00 00 0C 4A B3 BE
at.atTable.atEntry.atNetAddress.1.1.132.92.122.17 = IpAddress: 132.92.122.17
at.atTable.atEntry.atNetAddress.1.1.132.92.122.18 = IpAddress: 132.92.122.18
at.atTable.atEntry.atNetAddress.1.1.132.92.122.19 = IpAddress: 132.92.122.19
at.atTable.atEntry.atNetAddress.1.1.132.92.122.30 = IpAddress: 132.92.122.30
```

Dengan perintah arp –a di mesin khnemu, kita dapatkan informasi hostname, IP address dan alamat fisik interface:

```
khnemu # arp -a
weres.Cnrg.net (132.92.122.17) at 0:80:c8:3f:9e:15
tiet.Cnrg.net (132.92.122.18) at 0:0:c:75:b0:af
khnemu.Cnrg.net (132.92.122.19) at 0:80:48:e4:11:6 permanent
benu.Cnrg.net (132.92.122.30) at 0:0:c:4a:b3:be
```

7.8.4. Group ip

Group **ip** mendefinisikan banyak variabel dan tiga buah tabel, seperti diuraikan dalam tabel berikut.

Tabel 7.10 Variabel-variabel dalam group ip

Nama	**Tipe data**	**R/W**	**Keterangan**
ipForwarding	[1..2]	*	1= sistem meneruskan datagram IP, 2 = sistem tidak meneruskan.
ipDefaultTTL	INTEGER	*	Nilai TTL default jika lapisan transport tidak menyediakan nilai tersebut.
ipInReceives	Counter		Jumlah total datagram IP yang diterima dari semua interface
ipInHdrErrors	Counter		Jumlah datagram IP yang dibuang karena header rusak (misalnya checksum salah, nomor versi tidak cocok, TTL terlampaui)
ipInAddrErrors	Counter		Jumlah datagram IP yang dibuang karena alamat tujuan salah.
ipForwDatagrams	Counter		Jumlah datagram IP yang diupayakan untuk diteruskan.
ipInUnknownProtos	Counter		Jumlah datagram IP yang dialamatkan ke lokal dengan field protokol yang salah.
ipInDiscards	Counter		Jumlah datagram IP diterima yang dibuang karena habisnya ruang buffer

Nama	Tipe data	R/W	Keterangan
ipInDelivers	Counter		Jumlah datagram IP yang dikirim ke modul protokol yang sesuai.
ipOutRequests	Counter		Jumlah total datagram IP yang dilewatkan ke IP untuk pengiriman. Tidak termasuk yang dihitung oleh ipForwDatagrams.
ipOutNoRoutes	Counter		Jumlah datagram IP yang dibuang karena route tidak ditemukan.
ipReasmTimeout	INTEGER		Jumlah detik maksimum bahwa fragment yang diterima ditahan selama menunggu penyusunan kembali.
ipReasmReqds	Counter		Jumlah datagram IP diterima yang perlu disusun ulang.
ipReasmOKs	Counter		Jumlah datagram IP yang berhasil disusun ulang.
ipReasmFails	Counter		Jumlah kegagalan karena algoritma penyusunan IP.
ipFragOKs	Counter		Jumlah datagram IP yang berhasil difragmentasi
ipFragFails	Counter		Jumlah datagram IP yang perlu difragmentasi tetapi tidak bisa dilakukan ka-

Nama	Tipe data	R/W	Keterangan
			rena ada flag "jangan difragmentasi"
ipFragCreates	Counter		Jumlah fragment IP yang dihasilkan oleh fragmentasi
ipRoutingDiscards	Counter		Jumlah entri routing yang dipilih untuk dibuang meskipun mereka valid.

Tabel pertama dalam group `ip` adalah tabel IP address. Tabel ini berisi sebuah baris untuk tiap IP address dalam sistem. Setiap baris terdiri atas lima variabel, seperti ditunjukkan oleh tabel berikut:

Tabel 7.11 Tabel IP address: ipAddrTable

Tabel IP address, index = *<ipAdEntAddr>*

Nama	Tipe data	R/W	Keterangan
ipAdEntAddr	IpAddress		IP address untuk baris ini.
ipAdEntIfIndex	INTEGER		Nomor interface yang bersesuaian: ifIndex
ipAdEntNetMask	IpAddress		Subnet mask untuk IP address ini.
ipAdEntBcastAddr	[0..1]		Nilai bit least-significant dari IP address broadcast. Normalnya 1.
ipAdEntReasmMaxSize	[0..65535]		Ukuran datagram IP terbesar yang diterima oleh interface ini yang dapat disusun ulang.

Kita gunakan `snmpwalk` untuk mengakses seluruh tabel IP address:

```
khensu:~$ snmpwalk localhost inipassword ip.ipAddrTable
```

Keluaran dari perintah di atas setelah dihilangkan bagian ip.ipAddrTable.ipAddrEntry dari setiap obyek kejadian adalah:

```
ipAdEntAddr.0.0.0.0 = IpAddress: 0.0.0.0
ipAdEntAddr.44.132.16.1 = IpAddress: 44.132.16.1
ipAdEntAddr.44.132.80.1 = IpAddress: 44.132.80.1
ipAdEntAddr.127.0.0.1 = IpAddress: 127.0.0.1
ipAdEntAddr.132.92.121.114 = IpAddress: 132.92.121.114
ipAdEntIfIndex.0.0.0.0 = 3
ipAdEntIfIndex.44.132.16.1 = 2
ipAdEntIfIndex.44.132.80.1 = 9
ipAdEntIfIndex.127.0.0.1 = 1
ipAdEntIfIndex.132.92.121.114 = 4
ipAdEntNetMask.0.0.0.0 = IpAddress: 0.0.0.0
ipAdEntNetMask.44.132.16.1 = IpAddress: 255.0.0.0
ipAdEntNetMask.44.132.80.1 = IpAddress: 255.255.255.0
ipAdEntNetMask.127.0.0.1 = IpAddress: 255.0.0.0
ipAdEntNetMask.132.92.121.114 = IpAddress: 255.255.255.224
ipAdEntBcastAddr.0.0.0.0 = 0
ipAdEntBcastAddr.44.132.16.1 = 1
ipAdEntBcastAddr.44.132.80.1 = 1
ipAdEntBcastAddr.127.0.0.1 = 1
ipAdEntBcastAddr.132.92.121.114 = 1
```

Nilai IP address dan subnet mask dapat dibandingkan dengan nilai keluaran perintah `ifconfig` (hanya diperlihatkan baris pertama dan kedua bagi setiap interface):

```
khensu:~$ ifconfig
lo      Link encap:Local Loopback
        inet addr:127.0.0.1  Bcast:127.255.255.255  Mask:255.0.0.0
ed0     Link encap:10Mbps Ethernet  HWaddr 00:80:48:A8:C0:90
        inet addr:132.92.121.114 Bcast:132.92.121.127 Mask:255.255.255.224
```

```
sl0   Link encap:AMPR AX.25  HWaddr YC1DAV-1
      inet addr:44.132.80.1 Bcast:44.132.80.255 Mask:255.255.255.0
sl1   Link encap:AMPR AX.25  HWaddr YC1DAV-2
      inet addr:44.132.16.1 Bcast:44.132.16.255 Mask:255.255.255.0
```

Tabel berikutnya adalah tabel IP routing, seperti diuraikan dalam tabel berikut.

Tabel 7.12 Tabel IP routing: ipRouteTable

Tabel IP routing, index = <*ipRouteDest*>

Nama	Tipe data	R/W	Keterangan
ipRouteDest	IpAddress	*	IP address tujuan. Nilai 0.0.0.0 artinya entri default.
ipRouteIfIndex	INTEGER	*	Nomor interface: ifIndex.
ipRouteMetric1	INTEGER	*	Metrik routing primer. Maksud dari metrik tergantung dari protokol routing (ipRouteProto). Nilai –1 artinya tidak digunakan.
ipRouteMetric2	INTEGER	*	Metrik routing alternatif.
ipRouteMetric3	INTEGER	*	Metrik routing alternatif.
ipRouteMetric4	INTEGER	*	Metrik routing alternatif.
ipRouteNextHop	IpAddress	*	IP address dari router selanjutnya.
ipRouteType	INTEGER	*	Tipe route: 1= lainnya, 2 = route invalid, 3 = lansung, 4 = tidak langsung.

Tabel IP routing, index = *<ipRouteDest>*

Nama	Tipe data	R/W	Keterangan
ipRouteProto	INTEGER		Protokol routing: 1 = lainnya, 4 = ICMP redirect, 8 =RIP, 13 = OSPF, 14 = BGP, dan lainnya.
ipRouteAge	INTEGER	*	Jumlah detik sejak route terakhir diperbaharui atau diputuskan benar.
ipRouteMask	IpAddress	*	Mask yang secara logik di-AND-kan dengan IP address tujuan sebelum dibandingkan dengan ipRouteDest.
ipRouteMetric5	INTEGER	*	Metrik routing alternatif.
ipRouteInfo	ObjectID		Rujukan ke definisi MIB tertentu bagi protokol routing ini.

Tabel 7.13 adalah tabel address translation yang menggantikan group at.

Tabel 7.13 Tabel IP address translation: ipNetToMediaTable

Tabel IP address translation, index = *<ipNetToMediaIfIndex>.<ipNetToMediaNetAddress>*

Nama	Tipe data	R/W	Keterangan
ipNetToMediaIfIndex	INTEGER	*	Interface yang bersesuaian: ifIndex.

ipNetToMediaPhysAddress	PhysAddress	*	Alamat fisik.
ipNetToMediaNetAddress	IpAddress	*	IP address
ipNetToMediaType	[1..4]	*	Tipe pemetaan: 1 = lainnya, 2 = tidak divalidasi, 3 = dinamik, 4 = statik.

7.8.5. Group icmp

Grop `icmp` memiliki empat counter umum (jumlah total pesan ICMP input dan output, dan pesan ICMP input dan output dengan error) dan 22 counter untuk tipe pesan ICMP yang berbeda: 11 counter input dan 11 counter output. Lihat tabel berikut:

Tabel 7.14 Variabel dalam group icmp

Nama	Tipe data	R/W	Keterangan
icmpInMsgs	Counter		Jumlah total pesan ICMP yang diterima.
icmpInErrors	Counter		Jumlah pesan ICMP diterima yang rusak (misalnya checksum ICMP salah).
icmpInDestUnreachs	Counter		Jumlah pesan ICMP diterima yang tujuannya tidak dapat dihubungi.
icmpInTimeExcds	Counter		Jumlah pesan ICMP diterima yang waktunya terlampaui
icmpInParmProbs	Counter		Jumlah pesan ICMP diterima yang

Nama	Tipe data	R/W	Keterangan
			parameter-nya bermasalah.
icmpInSrcQuenchs	Counter		Jumlah pesan ICMP diterima yang sumbernya mengalami kongesti.
icmpInRedirects	Counter		Jumlah pesan ICMP diterima yang dikirimkan ulang.
icmpInEchos	Counter		Jumlah pesan ICMP diterima yang berisi echo request.
icmpInEchoReps	Counter		Jumlah pesan ICMP diterima yang berisi echo reply.
icmpInTimestamps	Counter		Jumlah pesan ICMP diterima yang berisi time-stamp request.
icmpInTimestampReps	Counter		Jumlah pesan ICMP diterima yang berisi timestamp reply.
icmpInAddrMasks	Counter		Jumlah pesan ICMP diterima yang berisi address mask request.
icmpInAddrMaskReps	Counter		Jumlah pesan ICMP diterima yang berisi address mask reply.
icmpOutMsgs	Counter		Jumlah total pesan ICMP yang dikirim.

Nama	Tipe data	R/W	Keterangan
icmpOutErrors	Counter		Jumlah pesan ICMP dikirim yang rusak (misal: checksum ICMP salah).
icmpOutDestUnreachs	Counter		Jumlah pesan ICMP dikirim yang tujuannya tidak dapat dihubungi.
icmpOutTimeExcds	Counter		Jumlah pesan ICMP dikirim yang waktunya terlampaui
icmpOutParmProbs	Counter		Jumlah pesan ICMP dikirim yang parameternya bermasalah.
icmpOutSrcQuenchs	Counter		Jumlah pesan ICMP dikirim yang sumbernya mengalami kongesti.
icmpOutRedirects	Counter		Jumlah pesan ICMP dikirim yang dikirimkan ulang.
icmpOutEchos	Counter		Jumlah pesan ICMP dikirim yang berisi echo request.
icmpOutEchoReps	Counter		Jumlah pesan ICMP dikirim yang berisi echo reply.
icmpOutTimestamps	Counter		Jumlah pesan ICMP dikirim yang berisi timestamp request.

Nama	Tipe data	R/W	Keterangan
icmpOutTimestampReps	Counter		Jumlah pesan ICMP dikirim yang berisi timestamp reply.
icmpOutAddrMasks	Counter		Jumlah pesan ICMP dikirim yang berisi address mask request.
icmpOutAddrMaskReps	Counter		Jumlah pesan ICMP dikirim yang berisi address mask reply.

7.8.6. Group tcp

Group tcp memiliki satu tabel, tabel TCP connection. Setiap hubungan (connection) ditampilkan dalam satu baris tabel yang berisi lima variabel: status hubungan, IP address lokal, nomor port lokal, IP address remote, dan nomor port remote. Variabel-variabel dalam group tcp ditunjukkan oleh tabel berikut.

Tabel 7.15 Variabel dalam group tcp

Nama	Tipe data	R/W	Keterangan
tcpRtoAlgorithm	INTEGER		Algoritma yang digunakan untuk menghitung nilai timeout retransmisi: 1 = tidak satu pun, 2 = RTU konstan, 3 = MIL-STD-1778 Appendix B, 4 = Van Jacobson.
tcpRtoMin	INTEGER		Nilai timeout retransmisi minimum, dalam mili detik.

Nama	Tipe data	R/W	Keterangan
tcpRtoMax	INTEGER		Nilai timeout retransmisi maksimum, dalam mili detik.
tcpMaxConn	INTEGER		Jumlah hubungan TCP maksimum. Nilai –1 jika dinamik.
tcpActiveOpens	Counter		Jumlah transmisi dari status CLOSED ke SYN_SENT.
tcpPasiveOpens	Counter		Jumlah transmisi dari status LISTEN ke SYN_RCVD.
tcpAttemptFails	Counter		Jumlah transmisi dari status SYN_SENT atau SYN_RCVD ke CLOSED, ditambah jumlah transmisi dari SYN_RCVD ke LISTEN.
tcpEstabResets	Counter		Jumlah transisi dari status ESTABLISED atau CLOSE_WAIT ke CLOSED.
tcpCurrEstab	Gauge		Jumlah hubungan saat ini dalam status ESTABLISED atau CLOSE_WAIT.
tcpInSegs	Counter		Jumlah total segmen yang diterima.
tcpOutSegs	Counter		Jumlah total segmen yang dikirim, termasuk yang hanya berisi byte retansmisi.
tcpRetransSegs	Counter		Jumlah total segmen retransmisi.
tcpInErrs	Counter		Jumlah total segmen yang diterima dengan kerusakan (misal

Nama	Tipe data	R/W	Keterangan
			cheksum yang salah)
tcpOutRsts	Counter		Jumlah toal segmen yang dikirim dengan flag RST diset.

Tabel 7.16 Tabel hubungan TCP: tcpConnTable

index =
<tcpConnLocalAddress>.<tcpConnLocalPort>.<tcpConnRemAddress>.<tcpConnRemPort>

Nama	Tipe data	R/W	Keterangan
tcpConnState	[1..12]	*	Status hubungan: 1 = CLOSED, 2 = LISTEN, 3 = SYN_SENT, 4 = SYN_RCVD, 5 = ESTABLISHED, 6 = FIN_WAIT_1, 7 = FIN_WAIT_2, 8 = CLOSE_WAIT, 9 = LAST_ACK, 10 = CLOSING, 11 = TIME_WAIT, 12 = hapus TCB. Manajer hanya bisa meset variabel ini ke nomor 12 (misal putuskan hubungan segera).
tcpConnLocalAddress	IpAddress		IP address lokal. 0.0.0.0 artinya agen akan menerima hubungan dari interface manapun.
tcpConnLocalPort	[0..65535]		Nomor port lokal.
tcpConnRemAddress	IpAddress		IP address remote.
tcpConnRemPort	[0..65535]		Nomor port remote.

Kita dapat menggunakan `snmpget` untuk mengetahui nilai beberapa variabel di atas. Misal dari mesin Linux, kita akses agen snmp yang ada di mesin Khnemu.

```
khensu:~$ snmpget khnemu public tcp.tcpRtoAlgorithm.0
tcp.tcpRtoMin.0 tcp.tcpRtoMax.0 tcp.tcpMaxConn.0
tcp.tcpRtoAlgorithm.0 = vanj(4)
tcp.tcpRtoMin.0 = 1000
tcp.tcpRtoMax.0 = 64000
tcp.tcpMaxConn.0 = -1
```

Mesin khnemu dengan sistem operasi FreeBSD 2.2.2 ternyata menggunakan algoritma Van Jacobson dalam agen SNMPnya, dengan timeout antara 1 detik sampai 64 detik, serta jumlah koneksi TCP yang tak terbatas.

7.9. Contoh Aplikasi: Menghitung Ulitisasi Link/Segment

Menghitung utilisasi link dapat digunakan untuk mengisolasi masalah unjuk kerja atau membantu menghindari kongesti melalui kemampuan perencanaan jangka panjang. Utilisasi link yang akan mempengaruhi unjuk kerja jaringan tergantung pada banyak faktor termasuk protokol data link di bawahnya, algoritma retransmisi, dan aplikasi yang memanfaatkan link tersebut. Karena variabel-variabel ini, beberapa organisasi penyedia atau yang mengimplementasikan jaringan Internet, akan memiliki utilisasi link yang berbeda yang mengakibatkan berkurangnya unjuk kerja yang dirasakan oleh pemakai. Unjuk kerja yang buruk ini sering ditandai oleh waktu respon yang lama.

Dengan menggunakan objek dari group MIB *interfaces* dapat dicari persentase utilisasi untuk sebuah perangkat

dalam media broadcast (seperti segmen Ethernet). Objek yang sama juga bisa digunakan untuk mengukur utilisasi pada media *full-duplex point-to-point* (seperti HDLC, PPP, dan sebagainya).

Objek *ifInOctets* dan *ifOutOctets* memberi informasi jumlah total byte yang diterima dan dikirim pada sebuah interface. Dengan menghitung *delta* untuk bilangan ini dan membaginya dengan bandwidth, akan diperoleh persentase utilisasi. *Bandwidth* dalam kilobit per detik dari interface ditemukan dalam objeck *ifSpeed*.

Rumus untuk menghitung utilisasi interface Ethernet (half duplex) adalah:

```
utilisasi = ( 8 * (delta(ifInOctets, t1, ifInOctets, t0) +
               delta(ifOutOctets, t1, ifOutOctets, t0))/ (t1 - t0)
            )/ifSpeed
```

Perkalian dengan 8 diperlukan untuk mengubah unit *ifInOctets* dan *ifOutOctets* (byte) ke unit *ifSpeed* (bit). Perlu diperhatikan, rumus ini hanya untuk menghitung utilisasi interface bukan utilisasi seluruh media.

Untuk media full-duplex point-to-point, kita perlu mengubah rumus agar hanya menggunakan delta input atau output maksimum. Jika tidak, maka yang terhitung adalah 200 % utilisasi (full bandwidth pada dua arah bersamaan). Rumus di bawah dapat dipakai untuk situasi ini.

```
utilisasi = ( 8 * max(delta(ifInOctets, t1, ifInOctets, t0) ,
                  delta(ifOutOctets, t1, ifOutOctets, t0))/ (t1 - t0)
            )/ifSpeed
```

Dengan melihat objek *ifType*, dapat diketahui rumus mana yang perlu dipakai untuk interface tertentu. Misalnya untuk interface full-duplex, rumus kedua

berlaku untuk ifType: 2(regular1822), 3(hd1822), 4(ddn-x25), 5(rfc877-x25), 16(lapb), 17(sdlc), 18(ds1), 19(e1), 20(basicISDN), 21(primaryISDN), 22(proprietaryPointTo PointSerial), 23(ppp), 28(slip), 30(ds3), 31(smds), dan 32(frame-relay).

Software manajemen jaringan seperti Scotty, yang dibuat dalam bahasa pemrograman TCL/TK, merupakan salah satu contoh yang sederhana dan mudah untuk mewujudkan aplikasi di atas. Kita dapat menulis sebuah script dalam bahasa TCL/TK dan dijalankan di dalam program Scotty ini. Script yang telah dibuat oleh penulis bisa dilihat di **http://netmon.itb.ac.id/~ismail/project/map-generator**, dalam file *utility.tcl*. Salah satu contoh keluaran dari script untuk menghitung utilisasi link ditunjukkan oleh gambar berikut.

SNMP-Statistics Report

File Options

Tue Jun 03 20:10:26 GMT+0700 1997 by netmon.itb.ac.id

Index	ifType	ifSpeed (bps)	Utility (%)	peakUtil (%)	Router
1	ethernet-csmacd	10000000	0.6660	0.8008	cisco-en.ITB
3	propPointToPointSerial	56000	95.4149	99.9895	cisco-en.ITB
1	ethernet-csmacd	10000000	0.0283	0.1277	ywcn-lembong
2	propPointToPointSerial	1544000	0.1232	0.6806	ywcn-lembong
3	propPointToPointSerial	1544000	0.0463	0.0853	ywcn-lembong
1	ethernet-csmacd	10000000	0.0508	0.1555	YWCN.Malang
2	propPointToPointSerial	1544000	0.1235	0.6812	YWCN.Malang
1	ethernet-csmacd	10000000	0.0140	0.0189	YWCN.Surabaya
2	propPointToPointSerial	1544000	0.0465	0.0842	YWCN.Surabaya
1	ethernet-csmacd	10000000	52.2525	53.2129	ciscol.nara
3	fddi	100000000	0.7092	1.0950	ciscol.nara
4	propPointToPointSerial	1536000	46.0133	66.7122	ciscol.nara

Gambar 7.7 Contoh program SNMP untuk menghitung utilitas interface

7.10. Ringkasan

Manajemen jaringan internet akan semakin diperlukan oleh seorang manajer jaringan atau administrator jaringan, manakala jaringan yang harus ditanganinya sudah semakin besar. Ketersediaan hubungan atau *availability of connection* setiap saat merupakan sebuah goal yang harus dicapai. Untuk itu, seorang administrator dituntut untuk mengetahui kondisi kesehatan dari jaringannya secepat mungkin, sebelum para pengguna jaringan mengetahui adanya masalah seperti *connection down*, paket yang didiscard tinggi, dan sebagainya.

Dengan SNMP, yang terdiri atas tiga elemen: agent, manajer, dan MIB, proses monitoring jaringan dapat dilakukan secara terpusat. Agen yang dipasang tersebar di setiap router atau host-host penting, akan memberitahu kepada sebuah manajer mengenai kondisi mesin yang di monitornya.

Pada prinsipnya, setiap device jaringan dapat dimonitor menggunakan SNMP. Yang diperlukan untuk keperluan tersebut adalah agen dan MIB. Banyak software agen SNMP yang dibuat menurut kebutuhan device, dengan informasi dalam MIB yang spesifik.

Sebuah jaringan komputer, secara umum memiliki objek-objek yang dapat diamati yang dikelompokkan ke dalam group-group: system, interfaces, at, ip, icmp, dan tcp. Untuk memperoleh informasi yang dibutuhkan misal utilitas interface, software manajer harus mengumpulkan informasi tentang beberapa objek yang relevan, kemudian dimasukkan ke dalam rumus yang sesuai untuk informasi utilitas tersebut.

Bab 8
Protokol Aplikasi TCP/IP

Lapisan paling atas dari TCP/IP adalah lapisan aplikasi. Pada Bab 7, kita telah mempelajari SNMP, yang sebenarnya juga salah satu protokol yang berjalan di lapisan aplikasi. Pembahasan tentang SNMP memang dipisahkan, mengingat cukup luas dan pentingnya SNMP. Seperti telah Anda ikuti pada Bab 7, SNMP menjadi salah satu bahasan tersendiri mengenai Internet, yang melahirkan satu bidang tersendiri yaitu Manajemen Jaringan.

Pada bab ini akan dibahas aplikasi-aplikasi TCP/IP yang lain, yaitu FTP, SMTP, dan HTTP. FTP atau *File Transfer Protocol* banyak digunakan untuk keperluan pengambilan file dari komputer satu ke komputer lain mulai dari ukuran yang kecil sampai yang besar. SMTP atau *Simple Mail Transport Protocol* berperan sebagai tukang pos yang mengirimkan pesan para pengguna Internet ke pada pengguna yang dituju. File juga bisa dikirim menggunakan SMTP ini, namun ukurannya relatif lebih kecil. HTTP atau *Hypertext Transfer Protocol* sering kita ketikkan sewaktu kita menggunakan program aplikasi browser web seperti Netscape atau Internet Explorer. Misal **http://www.yahoo.com**.

8.1. File Transfer Protocol

FTP atau *File Transfer Protocol* merupakan salah satu aplikasi TCP/IP yang banyak digunakan untuk memindahkan atau mengcopy file dari komputer satu ke komputer lainnya. Aplikasi ini adalah aplikasi yang telah dikembangkan sejak awal perkembangan Internet. Hal ini terlihat dari mulai didefinisikannya protokol ini sejak Internet menggunakan RFC sebagai alat standarisasi. Kata FTP sendiri telah muncul di RFC 172 yang diterbitkan tahun 1971.

Operasi protokol FTP ini cukup sederhana. Dengan menggunakan client FTP, seorang pengguna dapat melihat isi direktori, memindahkan file dari dan ke server FTP, serta membuat dan menghapus direktori di server tersebut. Dalam melakukan operasi yang berhubungan dengan pengiriman isi file, FTP menggunakan koneksi TCP tambahan yang khusus untuk mengirim isi file. Sekarang kita akan melihat secara sederhana proses yang terjadi di dalam FTP sewaktu kita melakukan transfer file.

8.1.1. Model Protokol FTP

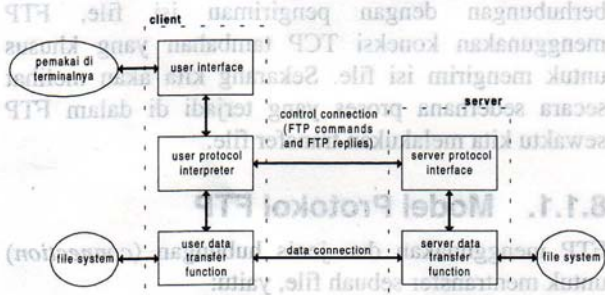
FTP menggunakan dua jenis hubungan (*connection*) untuk mentransfer sebuah file, yaitu:

- *Control connection*; yang digunakan pada pola hubungan antara client – server yang normal. Server membuka diri secara pasif di sebuah port khusus (*well-known port*) yaitu port 21. Selanjutnya server menunggu hubungan yang akan dilakukan oleh client. Client secara aktif membuka port 21 untuk membangun *control connection*.

Control connection ini akan dipertahankan sepanjang waktu selama client masih berkomunikasi dengan server. Hubungan ini digunakan oleh client untuk mengirim perintah-perintah ke server, dan server menggunakannya untuk memberi respon.

Hubungan ini bersifat "mengurangi delay", karena perintah-perintah biasanya diketik oleh manusia yang tentunya butuh kecepatan respon yang tinggi.

- *Data connection*; yang dibangun setiap kali sebuah file ditransfer antara client – server. Hubungan ini bersifat "memaksimalkan ukuran data yang ditransfer (*throughput*)", karena hubungan ini untuk transfer file.

Gambar 8.1 berikut memperlihatkan susunan client dan server serta dua hubungan diantara mereka.

Gambar 8.1 Model sebuah hubungan FTP

Pada model di atas, pemakai di terminalnya melakukan aktivitas FTP, melalui *user interface* baik yang berupa program windows FTP, atau yang *command line*. *User-protocol interpreter* selanjutnya yang akan melakukan hubungan *control connection* ke server. Perintah-perin-

tah FTP yang standard dikeluarkan oleh interpreter ini kepada server melalui hubungan *control connection*.

Gambar tersebut juga memperlihatkan ada dua interpreter protokol yang menangani kedua fungsi transfer data ketika dibutuhkan.

8.1.2. Representasi Data

Transfer file hanya dilakukan melalui *data connection*. Sedangkan *control connection* digunakan untuk mentransfer perintah-perintah dan balasan. Bagaimana file ditransfer dan disimpan telah disebutkan dalam spesifikasi protokol FTP dan ada banyak pilihan cara. Setiap pilihan harus dilakukan pada masing-masing dari keempat dimensi berikut:

Tipe File

ASCII (Default)
File teks ditransfer melalui *data connection* dalam NVT ASCII. Pengirim harus mengubah file teks lokal ke dalam NVT ASCII, dan sebaliknya penerima harus mengubah NVT ASCII ke tipe file teks lokal.

EBCDIC
Tipe ini merupakan alternatif lain untuk mentransfer file teks

BINARY (Image)
Data dikirim dalam bentuk aliran bit-bit yang terus-menerus. Biasanya untuk mentransfer file-file biner.

LOCAL
Merupakan cara untuk mentransfer file biner antara host-host yang memiliki ukuran byte yang berbeda. Jumlah bit dalam tiap byte ditentukan oleh pengirim.

Kontrol format, untuk tipe file ASCII dan EBCDIC

Nonprint (Default)
File tidak mengandung informasi format vertikal.

Kontrol format Telnet
File mengandung kontrol format vertikal Telnet untuk dibaca oleh printer.

Kontrol Fortran carriage
Karakter pertama setiap baris merupakan karakter kontrol format Fortran.

Struktur

Struktur File (Default).
File dianggap sebagai aliran byte-byte yang berurutan. Tidak ada struktur internal dari file.

Struktur Record
Struktur ini hanya digunakan oleh file teks (ASCII atau EBCDIC).

Struktur Page
Setiap halaman dikirimkan dengan nomor halaman sehingga penerima dapat menyimpan halaman-halaman tersebut secara random.

Mode transmisi

Ini menentukan bagaimana cara file ditransfer melalui *data connection*.

Mode Stream (Default)
File ditransfer sebagai aliran (stream) byte.

Mode Block
File ditransfer sebagai urutan blok-blok, masing-masing didahului oleh satu atau lebih byte header.

Mode Compressed

Jika dihitung, akan ada 72 cara yang berbeda untuk mentransfer dan menyimpan sebuah file. Namun tidak semua perlu dilakukan, karena beberapa tidak didukung oleh sebagian besar penerapan. Pada umumnya, penerapan FTP pada Unix ditentukan hanya pada pilihan ini:

- Tipe: ASCII atau Image
- Kontrol Format: nonprint
- Struktur: Struktur File
- Mode transmisi: Mode stream

8.1.3. Perintah-perintah FTP

Perintah-perintah FTP merupakan karakter ASCII sebanyak 3 sampai 4 byte, dan menggunakan huruf besar. Keseluruhan ada 30 perintah, dan beberapa perintah yang umum digunakan diperlihatkan oleh Tabel 8.1.

Tabel 8.1 Perintah-perintah FTP yang umum

Perintah	Keterangan
ABOR	Hentikan (abort) perintah FTP dan transfer data sebelumnya
LIST *filelist*	Sebutkan daftar file atau direktori
PASS *password*	Password di server
QUIT	Keluar dari server
RETR *namafile*	Ambil sebuah file
STOR *namafile*	Kirim sebuah file
SYST	Tipe sistem dari server
TYPE *type*	Menentukan tipe file: A = ASCII, I = Image
USER *namauser*	Nama pemakai di server

8.1.4. Reply FTP

Balasan (*reply*) FTP berupa bilangan tiga digit dalam ASCII. Software selanjutnya harus tahu bagaimana cara mengartikan balasan yang berupa bilangan tersebut. Maksud dari digit pertama dan kedua dari kode balasan ditunjukkan oleh Tabel 6.2.

Tabel 8.2 Arti dari digit pertama dan kedua dari kode reply tiga digit

Reply	Keterangan
1yz	Aksi sedang dimulai, tetapi perlu balasan lain sebelum dapat mengirim perintah baru.
2yz	Sebuah perintah baru boleh dikirim.
3yz	Perintah telah diterima, tetapi perintah lain harus dikirim.
4yz	Aksi yang diminta gagal dilaksanakan, dan dapat dikirim perintah ulangan karena kegagalannya sementara
5xy	Perintah ditolak dan mohon tidak dikirim lagi.
X0z	Syntax errors
X1z	Information.
X2z	Connection. Balasan dari control connection atau data connection.
X3z	Authentication dan accounting. Balasan dari perintah login atau accounting.
X4z	Unspecified
X5z	Status filesystem

Contoh reply FTP dapat dilihat di bawah ini.

```
220 khensu FTP server (Version wu-2.4(4) Tue Apr 29 13:38:27
CDT 1997) ready.
Name (localhost:ismail): ftp
331 Guest login ok, send your complete e-mail address as password.
Password:
```

```
230-The response 'sdf' is not valid
230-Welcome, archive user! This is an experimental FTP server.
452 Error writing file
500 Syntax error (unrecognized command)
501 Syntax error (invalid arguments)
```

Biasanya, setiap perintah pada FTP akan diberi reply satu baris. Misal, perintah QUIT akan dibalas:

```
221 Gooodbye
```

8.1.6. Pengaturan hubungan (*connection*)

Ada tiga jenis pemakaian pada *data connection,* yaitu:

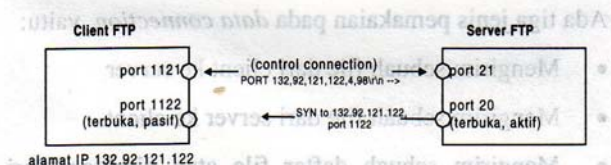
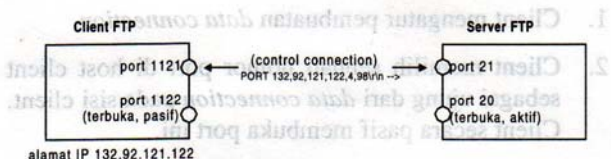
- Mengirim sebuah file dari client ke server
- Mengirim sebuah file dari server ke client.
- Mengirim sebuah daftar file atau direktori dari server ke client.

Prosedur normal untuk mentransfer file atau direktori adalah sebagai berikut:

1. Client mengatur pembuatan *data connection.*
2. Client memilih sebuah nomor port di host client sebagai ujung dari *data connection* pada sisi client. Client secara pasif membuka port ini.
3. Client mengirim nomor port ini ke server melalui *control connection* menggunakan perintah PORT.

Server menerima nomor port tersebut dari *control connection*, dan mengirim balasan secara aktif ke port di host client. Nomor port untuk *data connection* pada sisi server selalu 20.

Status hubungan sampai pada langkah ketiga ditunjukkan oleh Gambar 8.2. Kita anggap bahwa nomor port untuk *control connection* pada sisi client adalah 1121 dan nomor port untuk *data connection* adalah 1122. Client mengirim perintah PORT yang diikuti dengan alamat IP 132.92.121.1122 dan nomor port 16-bit. Nomor port 1122 = 4 x 256 + 98. Sehingga perintahnya adalah PORT 132,92,121,122,4,98.

Gambar 8.3 memperlihatkan status ketika server membuka *data connection*. Port di server untuk hubungan ini adalah port 20.

Gambar 8.2 Perintah PORT dikirim melalui control connection

Gambar 8.3 Server FTP membuka hubungan di data connection

8.1.7. Contoh aktivitas FTP

Berikut ini contoh aktivitas FTP yang disertai dengan keterangan pada setiap baris perintah dan reply. Perintah `ftp` pertama kali dijalankan di mesin *ns* yang bertindak sebagai client ke mesin *khensu* yang bertindak sebagai server.

Pilihan -d di belakang perintah `ftp` dimaksudkan agar perintah FTP yang dikirim client ke server ditampilkan di layar. Tanda `--->` menunjukkan teks setelahnya merupakan perintah yang dikirim oleh client ke server.

ns # ftp -d khensu	-d untuk debug
Connected to khensu	Control connection terbentuk
220 khensu FTP server (Version wu-2.4(4) Tue Apr 29 13:38:27 CDT 1997) ready.	Server merespon: SIAP
Name (khensu:admin): ismail	Client menampilkan promt login
---> USER ismail	Client mengirim nama pemakai
331 Password required for ismail.	Server minta password
Password:	Pemakai memasukkan password
---> PASS XXXX	Client mengirim password
230 User ismail logged in.	Server mengenal pemakai, oke
---> SYST	Client mengirim perintah SYST
215 UNIX Type: L8	Server memberitahu tipe sitem
Remote system type is UNIX.	
Using binary mode to transfer files.	
ftp> dir hear	Pemakai mengetik perintah DIR
---> PORT 132,92,121,122,156,65	Client mengirim perintah PORT

200 PORT command successful.	Server merespon: oke
---> LIST hear	Client mengirim perintah LIST
150 Opening ASCII mode data connection for /bin/ls.	Server menjawab perintah
-rw-r--r-- 1 ismail users 1701 Jan 1 1980 hear	
226 Transfer complete.	
ftp> quit	Pemakai mengetik perintah QUIT
---> QUIT	Client mengirim perintah QUIT
221 Goodbye	Server merespon, selamat tinggal

8.1.8. Anonymous FTP

Beberapa FTP server memberi kesempatan kepada pemakai yang tidak memiliki account untuk masuk dan memanfaatkan isi server. Fasilitas ini dikenal dengan **Anonymous FTP**. Ketika muncul prompt login seperti contoh di bawah, pemakai selanjutnya memasukkan kata **anonymous** atau **ftp**, dengan **alamat e-mail** sebagai password.

Berikut ini contoh anonymous FTP dengan login name **ftp**:

```
ns # ftp -d khensu
Connected to khensu
220 khensu FTP server (Version wu-2.4(4) Tue Apr 29
13:38:27 CDT 1997) ready.
Name (khensu:admin): ftp
---> USER ftp
331 Guest login ok, send your complete e-mail address as
password.
Password:
---> PASS XXXX
230 Guest login ok, access restrictions apply.
```

Berikut ini contoh anonymous FTP dengan login name **anonymous**:

```
ns # ftp -d khensu
Connected to khensu.
220 khensu FTP server (Version wu-2.4(4) Tue Apr 29
13:38:27 CDT 1997) ready.
Name (khensu:admin): anonymous
---> USER anonymous
331 Guest login ok, send your complete e-mail address as
password.
Password:
---> PASS XXXX
230-The response 'ismail' is not valid
230-Next time please use your e-mail address as your
password
230 Guest login ok, access restrictions apply.
```

Contoh di atas memperlihatkan bahwa jika kita memasukkan alamat e-mailnya salah, misal tidak ada tanda @, maka server akan memberitahu bahwa passwordnya salah. Namun, pemakai tetap bisa masuk dan memanfaatkan isi server FTP seperti biasa.

8.1.9. Direktori dan akses file

Perintah **ls** dapat dilakukan untuk mengetahui isi sebuah direktori. Contoh isi direktori yang dapat diakses oleh pemakai umum (*anonymous*) adalah:

```
ftp> ls
---> PORT 132,92,121,122,156,73
200 PORT command successful.
---> LIST
150 Opening ASCII mode data connection for /bin/ls.
total 128
drwxr-xr-x  7 root   root    16384 Mar  6 14:15 .
drwxr-xr-x  7 root   root    16384 Mar  6 14:15 ..
drwxr-xr-x  2 root   root    16384 Dec 20 05:32 bin
```

```
drwxr-xr-x  2 root   root     16384 Dec 20 05:32 etc
drwxrwxrwx  2 root   root    16384 Mar  6 14:15 incoming
drwxr-xr-x  2 root   root     16384 Dec 20 05:32 lib
drwxr-xr-x  3 root   root     16384 Feb 23 01:58 pub
-rwxr-xr-x  1 root   root       312 Dec 20 05:32
welcome.msg
226 Transfer complete.
```

Direktori tempat menyimpan segala dokumen dan file yang dapat diakses oleh pemakai anonymous adalah **pub** dan **incoming**. Bedanya, di direktori **pub**, pemakai hanya bisa membaca atau mengambil file saja, sedangkan di direktori **incoming**, pemakai bisa mengambil, menulis atau menyimpan dan menghapus file atau direktori.

Pemakai dapat mengambil (*download*) file dengan memberi perintah GET. Berikut ini contoh perintah GET.

```
ftp> ls
---> PORT 132,92,121,122,156,75
200 PORT command successful.
---> LIST
150 Opening ASCII mode data connection for /bin/ls.
total 15600
drwxr-xr-x  2 root   root     16384 Feb 23 02:01 .
drwxr-xr-x  3 root   root     16384 Feb 23 01:58 ..
-rw-r--r--  1 root   root    870400 Feb 23 01:59 ax25-
utils-2.0.12b.tar
-rw-r--r--  1 root   root  13785745 Feb 23 01:58 linux-
2.0.29.tar.Z
-rw-r--r--  1 root   root    624640 Feb 23 01:59 net-tools-
1.32-alpha.tar
226 Transfer complete.
ftp> bin                              Tipe file: BINER (image)
---> TYPE I
200 Type set to I.
ftp> hash
Hash mark printing on (1024 bytes/hash mark).
```

```
ftp> get ax25-utils-2.0.12b.tar
local: ax25-utils-2.0.12b.tar remote: ax25-utils-2.0.12b.tar
---> PORT 132,92,121,122,156,76
200 PORT command successful.
---> RETR ax25-utils-2.0.12b.tar
150 Opening BINARY mode data connection for ax25-utils-
2.0.12b.tar (870400 bytes
).
##################################################
################################
##################################################
################################
... dihapus ...
##################################################
################################
##################################################
226 Transfer complete.
870400 bytes received in 2.21 seconds (385.47 Kbytes/s)
```

Berikut ini contoh perintah untuk menyimpan (*upload*) file ke direktori **incoming**.

```
ftp> cd /incoming            Masuk ke direktori incoming
---> CWD /incoming
250 CWD command successful.
ftp> ascii         Tipe file yang diupload: TEXT/ASCII
---> TYPE A
200 Type set to A.
ftp> hash
Hash mark printing on (1024 bytes/hash mark).
ftp> put encap.txt                  Upload file
local: encap.txt remote: encap.txt
---> PORT 132,92,121,122,156,79
200 PORT command successful.
---> STOR encap.txt
150 Opening ASCII mode data connection for encap.txt.
#############################
226 Transfer complete.
29680 bytes sent in 0.06 seconds (476.04 Kbytes/s)
```

8.1.10. Tip mengakses file

Nama file yang pendek pada umumnya lebih mudah bagi kita untuk mengetikkannya. Namun, untuk nama file yang panjang, kemungkinan salah ketik akan cukup besar. Jika akses kita ke server FTP cukup cepat, tidak masalah. Tetapi jika aksesnya lambat, misal lewat dialup telepon, kesalahan ketik bisa menambah pulsa. Berikut ini contoh kesalahan dalam mengetik nama file yang menghasilkan error.

```
ftp> get ax25-utils-2.0.12.b.tar
local: ax25-utils-2.0.12.b.tar remote: ax25-utils-2.0.12.b.tar
---> PORT 132,92,121,122,156,85
200 PORT command successful.
---> RETR ax25-utils-2.0.12.b.tar
550 ax25-utils-2.0.12.b.tar: No such file OR directory.
```

Kita dapat menghindari hal di atas dengan mengetikkan perintah **mget** sebagai ganti perintah **get**, dan **mput** sebagai ganti perintah **put**, seperti diperlihatkan contoh di bawah.

```
ftp> mget ax*        Ambil semua file yang dimulai oleh ax
---> PORT 132,92,121,122,156,91
---> NLST ax*
mget ax25-utils-2.0.12b.tar? y
---> TYPE I
200 Type set to I.
---> PORT 132,92,121,122,156,92
200 PORT command successful.
---> RETR ax25-utils-2.0.12b.tar
150 Opening BINARY mode data connection for ax25-utils-
2.0.12b.tar (870400 bytes
).
226 Transfer complete.
870400 bytes received in 2.20 seconds (386.43 Kbytes/s)
mget ax25-utils-2.0.12b.tar.Z? n
```

8.2. E-mail dan SMTP (*Simple Mail Transport Protocol*)

Sebuah studi pada tahun 1991 menunjukkan bahwa separuh dari hubungan TCP adalah untuk *Simple Mail Transport Protocol* (SMTP). Walaupun persentasenya saat ini tentu turun karena perkembangan WWW yang cepat sejak 1994, aplikasi ini adalah salah satu layanan Internet yang paling banyak digunakan. E-mail terkenal karena memberikan cara yang mudah dan cepat dalam mengirim informasi. Selain itu juga dapat menangani catatan kecil sampai file yang ukurannya cukup besar. Dan tidak perlu diragukan lagi, bahwa sebagian besar pengguna mengirim file menggunakan e-mail dari pada menggunakan program transfer file. Rata-rata pesan dalam e-mail tidak mencapai sepuluh kilobyte dan beberapa pesan mengandung beberapa megabyte data, karena digunakan untuk mengirim file.

Gambar 8.4 memperlihatkan pertukaran e-mail menggunakan TCP/IP.

Gambar 8.4 Komponen konseptual sistem e-mail

Pemakai di terminalnya berhubungan dengan *user agent* (UA). Beberapa agent e-mail yang populer antara lain: Pine, Pegasus, dan Eudora. Pertukaran mail menggunakan TCP dilakukan oleh *Message Transport Agent* (MTA). MTA yang paling umum untuk Unix adalah *Sendmail*. Pemakai awam biasanya tidak berhubungan dengan MTA ini. Ini adalah tanggung jawab administrator untuk mengatur MTA lokal.

Pada bagian ini akan dipelajari pertukaran elektronik mail antar dua MTA menggunakan TCP. *User agent* tidak akan dibahas di sini.

8.2.1. Protokol SMTP

Komunikasi antara dua MTA menggunakan NVT ASCII. Perintah dikirim oleh client ke server, dan server merespon dengan kode balasan numerik dan beberapa string yang dapat dibaca. Hal ini mirip dengan FTP.

Perintah yang dapat dikirim client ke server jumlahnya sedikit, kurang dari selusin. Bandingkan dengan FTP yang memiliki lebih dari 40 perintah. Untuk menjelaskan perintah-perintah ini, lebih mudah jika dilakukan dengan contoh.

8.2.2. Contoh sederhana

Kita akan mengirim sebuah pesan satu baris dan akan kita lihat hubungan SMTPnya. Kita menjalankan *user agent* dengan tambahan *flag -v*, yang kemudian dikirim ke MTA (disini menggunakan Sendmail). MTA akan menampilkan apa yang dikirim dan diterimanya melalui hubungan SMTP. Baris yang dimulai dengan

>>> adalah perintah yang dikirim oleh client SMTP, dan baris yang dimulai dengan kode balasan 3 digit adalah dari server SMTP. Berikut ini adalah contoh session interaktifnya:

```
khensu $ mail -v ismail@hathor.cnrg.net
Subject: test
satu dua tiga
.
EOT
ismail@hathor.cnrg.net...
Connecting to hathor.cnrg.net. via esmtp...
220 hathor.cnrg.net SMTP HMEITB ready
>>> EHLO khensu.cnrg.net
500 Command unrecognized
>>> HELO khensu.cnrg.net
250 hathor.cnrg.net, Share and Enjoy!
>>> MAIL From:<ismail@khensu.cnrg.net>
250 Ok
>>> RCPT To:<ismail@hathor.cnrg.net>
250 Ok
>>> DATA
354 Enter mail, end with .
>>> .
>>> .
250 Sent
ismail@hathor.cnrg.net... Sent (Sent)
Closing connection to hathor.cnrg.net.
>>> QUIT
221 Closing
```

Untuk mengirim pesan e-mail, hanya ada lima perintah yang digunakan, yaitu: HELO, MAIL, RCPT, DATA, dan QUIT.

Pada contoh di atas, kita mengetik *mail* untuk menjalankan *user agent*. Selanjutnya kita diminta mengisi *Subject* dan *Body of the message* (isi pesan). Untuk mengakhiri pesan, kita mengetik sebuah titik

pada baris paling akhir, yang selanjutnya *user agent* akan mengirim mail tersebut ke MTA.

SMTP sangat sederhana. Komunikasi antara client dan server terdiri dari teks-teks yang mudah dibaca. Meski SMTP mendefinisikan perintah-perintah secara kaku, namun kita masih bisa dengan mudah membaca transkrip interaksi antara client dan server.

Mula-mula, client melakukan hubungan TCP secara aktif ke port 25, dan menunggu kode balasan 220 READY FOR MAIL, yaitu ucapan selamat datang dari server. Respon server ini harus dimulai dengan FQDN (*fully qualified domain name*) dari server, misal `hathor.cnrg.net`.

Selanjutnya client memperkenalkan dirinya dengan perintah EHLO, yaitu perintah yang ada pada ESMTP (akan dibahas selanjutnya). Pada contoh di atas, server memberi respon *command unrecongnized*, karena server menjalankan SMTP, bukan ESMTP. Selanjutnya client mengirim perintah HELO, yaitu perintah primitif yang ada pada SMTP versi awal. Argumen di belakang perintah tersebut harus FQDN dari client, misal khensu.cnrg.net

Server merespon dengan memberikan identitas dirinya kepada client. Jika komunikasi sudah terbentuk, client dapat mengirim lebih dari satu pesan, mengakhiri hubungan, atau meminta server untuk mengirim aturan bagi pengirim dan penerima, sehingga pesan dapat mengalir dengan arah yang sebaliknya.

Transaksi mail dimulai dengan perintah MAIL, yang menjelaskan siapa pengirim pesan ini. Server selanjutnya mempersiapkan struktur datanya agar dapat

menerima pesan baru, dan membalas perintah MAIL tersebut dengan kode 250, atau lengkapnya *250 OK*.

Perintah selanjutnya RCPT, menjelaskan siapa penerimanya. Jika ada banyak penerima, maka beberapa perintah RCPT dapat dikeluarkan. Server harus mengirim pemberitahuan bagi setiap perintah RCPT ini dengan mengirim respon *250 OK*, atau pesan kesalahan, misal *550 No such user here*.

Isi pesan dikirim oleh client dengan perintah DATA yang diakhiri dengan mengirim satu baris data yang hanya berisi satu titik. Server merespon dengan mengirim pesan 354 Start mail input dan menentukan urutan karakter tertentu yang dijadikan sebagai tanda akhir pesan e-mail. Urutan ini sebenarnya terdiri atas 5 karakter: *carriage return, line feed,* titik, *carriage return,* dan *line feed.*

QUIT dikirim terakhir untuk mengakhiri transaksi pengiriman pesan e-mail ini. Server meresponnya dengan mengirim pesan 221, yang berarti setuju untuk menghentikan transaksi. Kedua pihak akhirnya menutup hubungan TCP.

Di bawah ini adalah baris-baris e-mail yang diterima oleh hathor.cnrg.net:

```
Received:  by hathor.cnrg.net (8.8.5/8.8.5) id TAA02062
        for ismail@hathor.cnrg.net; Thu, 12 Mar 1998 19:04:22
+0700 (JAVT)
Date: Thu, 12 Mar 1998 19:04:22 +0700 (JAVT)
From: Ismail Fahmi <ismail@khensu.cnrg.net>
Message-Id: <199803121204.TAA02062@khensu.cnrg.net>
To: ismail@hathor.cnrg.net
Subject: test

satu dua tiga
```

Sebenarnya SMTP jauh lebih kompleks dibandingkan dengan yang dijelaskan di sini. Misalnya, jika seorang pemakai pindah, server bisa tahu dimana mailbox yang baru, dan memberi tahu client agar menggunakan alamat terbaru tersebut.

8.2.3. Komponen E-mail

Elektronik mail terdiri atas tiga komponen, yaitu:

Envelope, atau amplop. Ini digunakan oleh MTA untuk pengiriman. Dalam contoh sebelumnya, envelope ditandai dengan dua buah perintah SMTP:

```
MAIL From: <ismail@khnemu.cnrg.net>
RCPT To: <ismail@khensu.cnrg.net>
```

Isi dan interpretasi dari envelope SMTP ditentukan di RFC 821. RFC ini juga menentukan protokol yang digunakan untuk mengirim mail melalui hubungan TCP.

Header, digunakan oleh *user agent*. Dalam contoh sebelumnya, ada sembilan field header, yaitu: Received, Message-Id, From, Date, Reply-To, X-Phone, X-Mailer, To, dan Subject. Setiap field header berisi sebuah nama yang diikuti oleh sebuah titik dua (:), dan nilai dari field header tersebut. Format dan interpretasi atas field header ini ditentukan dalam RFC 822. Field header yang panjang, seperti Received, akan dilipat ke dalam beberapa baris, dengan ditambah sebuah spasi kosong di depannya.

Body merupakan isi pesan dari pengirim ke penerima. Dalam RFC 822 disebutkan bahwa body ini merupakan baris-baris dalam bentuk teks NVT ASCII. Setiap baris yang dikirim menggunakan perintah DATA, tidak boleh melebihi 1024 byte.

8.2.4. Relay Agent

Baris pertama informasi yang diberikan oleh MTA lokal pada contoh kita di atas adalah "*Connecting to* mailhost *via* ether". Pesan yang sudah diterima oleh MTA lokal dari *user agent* (UA) dikirim ke mailhost. Mailhost ini adalah host yang bertindak sebagai mesin relay untuk pengiriman mail. Jadi setiap mail yang dikirim ke luar jaringan, akan dikirim dulu ke relay agent ini, dan selanjutnya menjadi tanggung jawab relay agent untuk meneruskan ke tujuan.

Host yang bertindak sebagai relay agent harus didaftarkan di DNS sebagai MX (Mail Exchanger) dan setiap sistem e-mail di dalam domainnya diset agar mengirim mail mereka ke host ini.

Untuk host-host yang digunakan dalam buku ini, misal khensu.cnrg.net, relay agent-nya dapat dilihat menggunakan perintah `host khensu` seperti di bawah ini:

```
khnemu # host khensu
khensu.cnrg.net has address 132.92.122.3
khensu.cnrg.net has address 132.92.122.44
khensu.cnrg.net mail is handled (pri=20) by osiris.cnrg.net
khensu.cnrg.net mail is handled (pri=10) by khensu.cnrg.net
khensu.cnrg.net mail is handled (pri=100) by sekhmet.cnrg.net
```

Relay agent untuk khensu di set ke server *osiris* dan *sekhmet*. Server *osiris* merupakan prioritas kedua setelah *khensu* itu sendiri, dan terakhir adalah *sekhmet*. Informasi untuk host *osiris* adalah:

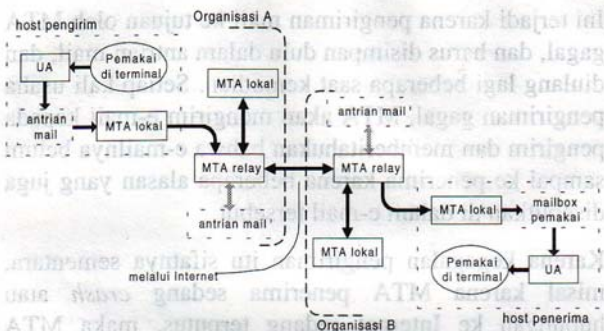
```
khnemu # host osiris
osiris.cnrg.net has address 132.92.121.33
osiris.cnrg.net has address 132.92.122.1
osiris.cnrg.net mail is handled (pri=100) by sekhmet.cnrg.net
osiris.cnrg.net mail is handled (pri=10) by isis.cnrg.net
```

Relay agent dari *osiris* juga diset ke *sekhmet*. Server *sekhmet* memiliki catatan DNS seperti di bawah ini.

```
khnemu # host sekhmet
sekhmet.cnrg.net has address 132.92.128.2
sekhmet.cnrg.net has address 132.92.128.5
sekhmet.cnrg.net has address 132.92.129.2
sekhmet.cnrg.net mail is handled (pri=10) by sekhmet.cnrg.net
```

Relay agent untuk *sekhmet* ini adalah dirinya sendiri.

Di Internet, sebagian besar organisasi telah menggunakan sistem relay. Dengan sistem ini, kita dapat menyembunyikan setiap sistem e-mail individual dari luar. Gambar 8.5 memperlihatkan sistem elektronik mail Internet yang menggunakan sistem relay di kedua ujungnya.

Ada empat MTA pada skenario tersebut, antara pengirim dan penerima. MTA lokal hanya meneruskan mail ke MTA relay milik organisasi yang sama. Dengan demikian, komunikasi antara kedua MTA ini menggunakan SMTP melalui *intranet* organisasi tersebut. MTA relay pada organisasi pengirim, meneruskan mail ke MTA relay pada organisasi penerima melalui Internet. MTA relay yang satunya ini selanjutnya mengirim mail ke host penerima, melalui komunikasi dengan MTA lokal pada host penerima tersebut. Semua MTA pada skenario ini menggunakan protokol SMTP.

Gambar 8.5 Elektronik mail Internet, dengan dua sistem relay pada kedua ujung

8.2.5. Interval Retry

Kita tentunya pernah menerima pesan seperti di bawah ini:

```
Date: Fri, 13 Mar 1998 13:08:59 +0700 (JVT)
From: Mail Delivery Subsystem <MAILER-
DAEMON@mx.other.net>
To: ismail@khnemu.cnrg.net
Subject: Warning: could not send message for past 4 hours
Parts/attachments:
  1   Shown    16 lines  Text
  2   Shown   377 bytes  Message
  3   Shown   2.4 KB     Message, "Re: buku"
  3.1 Shown    29 lines  Text
----------------------------------------
    ******************************************
    **     THIS IS A WARNING MESSAGE ONLY      **
    **  YOU DO NOT NEED TO RESEND YOUR MESSAGE  **
    ******************************************
The original message was received at Fri, 13 Mar 1998 08:40:52 +0700 (JVT)
from root@ns1.other.net [10.10.10.1]
.... dihapus....
```

Ini terjadi karena pengiriman mail ke tujuan oleh MTA gagal, dan harus disimpan dulu dalam antrian mail, dan diulang lagi beberapa saat kemudian. Setiap kali usaha pengiriman gagal, MTA akan mengirim e-mail kepada pengirim dan memberitahukan bahwa e-mailnya belum sampai ke penerima karena beberapa alasan yang juga disebutkan di dalam e-mail tersebut.

Karena kegagalan pengiriman itu sifatnya sementara, misal karena MTA penerima sedang *crash* atau hubungan ke Internet sedang terputus, maka MTA pengirim tidak boleh putus asa dan tetap harus mencoba sampai 4-5 hari.

8.2.6. SMTP versi mutakhir

Telah dilakukan perubahan pada elektronik mail Internet. Beberapa perintah SMTP terbaru ditambahkan dan berdampak pada envelope, karakter non-ASCII dapat digunakan dalam header, dan struktur ditambahkan ke body (MIME). Ekstension pada ketiga bagian mail Internet ini akan dijelaskan pada bagian ini.

8.2.7. Extended SMTP

Extended SMTP atau ESMTP merupakan hasil framework yang ditambahkan kepada SMTP. Adanya kemampuan baru pada ESMTP ini tidak mempengaruhi implementasi lama yang telah ada.

EHLO, adalah perintah pengganti HELO, pada ESMTP. Ini merupakan pemberitahuan kepada server bahwa client akan memanfaatkan kemampuan baru pada ESMTP. Jika client membuka hubungan dengan perintah HELO, berarti dia masih ingin menggunakan

implementasi yang lama, yaitu SMTP. Server yang kompatibel dengan ESMTP akan merespon dengan memberi kode 250. Respon ini normalnya bersifat multi baris yang setiap baris terdiri atas sebuah keyword dan argumen pilihan. Keyword-keyword ini menunjukkan ekstension SMTP yang didukung oleh server.

Inisialisasi hubungan kepada beberapa server SMTP yang mendukung ESMTP akan kita lihat, untuk mengetahui ekstension SMTP yang didukung oleh masing-masing server. Kita melakukannya dengan perintah Telnet ke port 25.

```
khnemu # telnet khensu 25
Trying 132.92.122.3...
Connected to khensu.cnrg.net.
Escape character is '^]'.
220 khensu.cnrg.net ESMTP Sendmail 8.8.5/8.8.5; Mon, 9 Mar
1998 20:48:21 +0700
ehlo khnemu
250-khensu.cnrg.net Hello khnemu.cnrg.net [], pleased to meet
you
250-EXPN
250-VERB
250-8BITMIME
250-SIZE
250-DSN
250-ONEX
250-ETRN
250-XUSR
250 HELP
```

Exteded command ditunjukkan dibelakang kode 250, yaitu EXPN, VERB, 8BITMIME, SIZE, DSN, ONEX, ETRN, XUSR, dan HELP. Server ESMTP ini menyatakan perintah-perintah pilihan dari RFC 821 yang didukungnya, dan dalam contoh ini adalah EXPN

dan HELP. Keyword 8BITMIME melarang client untuk mengirim karakter selain NVT ASCII. Keyword SIZE akan membolehkan client memasukkan ukuran mail (bytes) setelah perintah MAIL FROM. Setiap keyword yang dimulai dengan X, seperti XUSR, merujuk pada ekstension SMTP lokal.

Contoh berikut ini juga mendukung ESMTP. Perhatikan adanya perbedaan option dari RFC 821 yang didukung oleh server. Di sini, kita tidak temui EXPN.

```
khnemu # telnet sekhmet 25
Trying 132.92.128.5...
Connected to mx1.cnrg.net.
Escape character is '^]'.
220 mx1.cnrg.net ESMTP
ehlo khnemu
250-mx1.cnrg.net Hello khnemu.Cnrg.net [132.92.122.99],
pleased to meet you
250-8BITMIME
250-SIZE
250-DSN
250-ONEX
250-ETRN
250-XUSR
250 HELP
```

Contoh di bawah ini adalah ESMTP pada Mercury 2.30. Ditunjukkan di sini, hanya HELP dan SIZE yang didukungnya.

```
khnemu # telnet anubis 25
Trying 132.92.121.2...
Connected to alexandra.cnrg.net.
Escape character is '^]'.
220 alexandra.cnrg.net Mercury 1.30 SMTP server ready.
ehlo khnemu
250-alexandra.cnrg.net Hello khnemu; ESMTPs are:
250-HELP
250 SIZE 0
```

Contoh berikut adalah server SMTP yang dimiliki oleh NOS (Network Operating System), yaitu server elektronik mail untuk jaringan radio paket. Server ini tidak mendukung Extended SMTP, sehingga memberi pesan kesalahan *500 Command unrecognized*, terhadap perintah EHLO.

```
khnemu # telnet hathor 25
Trying 132.92.36.167...
Connected to hathor.cnrg.net.
Escape character is '^]'.
220 hathor.cnrg.net SMTP HMEITB ready
ehlo khnemu
500 Command unrecognized
rset
250 Reset state
```

Selanjutnya, client harus mengirim perintah RSET dan diikuti oleh perintah HELO.

8.2.8. Header: Karakter Non-ASCII

Karakter ASCII tidak mendukung penulisan karakter-karakter seperti β, ψ, atau karakter-karakter tambahan seperti pada huruf Latin. Karakter yang tidak ada dalam ASCII disebut karakter non-ASCII dan didefinisikan penulisannya pada header mail oleh RFC 1522.

Field header dapat berisi kata yang dikodekan. Formatnya adalah:

=? charset ? encoding ? encoded-text ?=

charset adalah spesifikasi set karakter. Nilai yang valid ada dua, yaitu: `us-ascii` dan `iso-8859-X`, dengan X berupa bilangan satu digit. *encoding* adalah satu huruf yang menyatakan metode pengkodean. Ada dua nilai untuk pengkodean ini:

Pengkodean Q, artinya *quoted-printable* yang ditujukan untuk kumpulan karakter Latin. Sebagian besar karakter dikirim sebagai NVT ASCII (bit tertinggi adalah 0). Karakter lainnya dengan ke delapan bit diset semua, dikirim dalam bentuk tiga karakter: pertama adalah karakter =, diikuti oleh dua digit hexadesimal. Contoh, karakter e' (hexadesimalnya 0xe9) dikirim sebagai tiga karakter =E9.

Pengkodean B, atau base-64. Tiga bytes teks berurutan (24 bits), dikodekan berdasarkan 6 bit dasar sebanyak 4 buah bit dasar. Selanjutnya, tiap 6 bit dasar tersebut direpresentasikan oleh salah satu dari 64 karakter NVT ASCII yang sesuai. Tabel di bawah ini memperlihatkan sebagian pengkodean 6 bit dasar oleh 64 NVT ASCII.

Nilai 6-bit	karakter ASCII	nilai 6-bit	karakter ASCII
0	A	10	Q
1	B	11	R
a	K	1a	a
b	L	1b	b

Jika jumlah karakter yang dikodekan bukan kelipatan 3, maka digunakanlah tanda *sama dengan* (=) sebagai karakter pengisinya.

8.2.9. Body: *Multipurpose Internet Mail Extension* (MIME)

Jika RFC 822 menentukan body message sebagai baris-baris teks dalam NVT ASCII tanpa struktur, maka RFC 1521 mendefinisikan ekstension yang memungkinkan adanya struktur dalam body message e-mail. Hal ini disebut MIME, Multipurpose Internet Mail Extension.

MIME tidak memerlukan ekstension pada ESMTP atau header non-ASCII. MIME hanya menambah beberapa header yang memberitahukan kepada penerima struktur dari body message. Dengan demikian, body masih bisa dikirim menggunakan NVT ASCII, tidak perduli isi dari mail. Kalau pada ESMTP ada perintah SIZE yang menyatakan panjang maksimum mail, sementara MIME bisa sangat panjang, maka ESMTP harus tidak berpengaruh pada MIME. Yang diperlukan agar MIME bisa diterapkan adalah, kedua ujung, baik pengirim maupun penerima, harus memiliki *user agent* yang mengerti MIME.

Ada lima field header pada MIME, yaitu:

- Mime-Version:
- Content-Type:
- Content-Transfer-Encoding:
- Content-ID:
- Content-Description:

Contoh, pada message mail Internet, akan ada dua baris header seperti berikut ini:

```
Mime-Version: 1.0
Content-Type: TEXT/PLAIN; charset=US-ASCII
```

Default dari Internet adalah MIME versi 1.0 dengan charset US-ASCII. MIME mendefinisikan tujuh Content-type. Tabel berikut ini memperlihatkan ke tujuh tipe dan sub-tipenya.

Content-Type	Subtype	Keterangan
text	plain	Teks yang tidak diformat
	richtext	Teks dengan format sederhana: tebal, miring, garis bawah, dan seterusnya.
	enriched	Penyederhanaan dan penyempurnaan dari richtext.
multipart	mixed	Beberapa bagian dari body diproses secara berurutan.
	parallel	Beberapa bagian dari body dapat diproses secara paralel
	digest	Sebuah digest elektronik mail
	alternative	Beberapa bagian dari body ada, semua menggunakan isi semantik yang identik.
message	rfc822	Content adalah pesan mail berdasarkan RFC 822.
	partial	Content adalah bagian dari sebuah pesan mail.
	external-body	Content merupakan penunjuk ke pesan yang sebenarnya.
application	octet-stream	Biasanya data biner.
	postscript	Sebuah program postscript.
image	jpeg	Format ISO 10918.
	gif	Format Graphic Interface Format dari CompuServe.
audio	basic	Pengkodean menggunakan format µ-law ISDN 8-bit.
video	mpeg	Format ISO 11172.

Pengkodean untuk transfer ditentukan oleh field header Content-Transfer-Encoding. Ada beberapa format pengkodean yang berbeda, yaitu:

Default, NVT ASCII (7 bit).
quoted-printable, seperti pada contoh sebelumnya. Ini berguna jika ada beberapa karakter yang menggunakan 8-bit set.

base64, seperti contoh sebelumnya.
8bit, yang mengandung karakter baris dan beberapa karakter non-ASCII.

binary, yaitu data 8-bit yang tidak mengandung baris.
Berikut ini contoh mail yang mengandung MIME dengan Content-Type MULTIPART dan subtype MIXED. Mail ini mengandung beberapa bagian atau attachment. Bagian pertama memiliki Content-Type TEXT dan bagian kedua adalah IMAGE.

```
Received: from localhost (ismail@localhost)
     by khnemu.cnrg.net (8.8.5/8.8.5) with SMTP id UAA04148
     for <ismail@khensu.cnrg.net>; Mon, 9 Mar 1998 20:54:13
+0700 (JAVT)
Date: Mon, 9 Mar 1998 20:54:13 +0700 (JAVT)
From: Ismail Fahmi <ismail@khnemu.cnrg.net>
To: ismail@khensu.cnrg.net
Subject: test MIME
Message-ID: <Pine.BSF.3.96.980309205335.4080B-
100001@khnemu.cnrg.net>
MIME-Version: 1.0
Content-Type: MULTIPART/MIXED; BOUNDARY="0-1995578357-
889451653=:4080"
Parts/attachments:
  1 Shown    3 lines  Text
  2 14 KB    Image, "Photo Adnan"
----------------------------------------


Test MIME
```

8.3. Hypertext Transfer Protocol

Sebelum ini telah dijelaskan bahwa Hypertext Transfer Protocol digunakan untuk jenis layanan WWW di jaringan TCP/IP. Spesifikasi protokol ini didefinisikan oleh Tim Berners-Lee dalam RFC 1945 dan digunakan di Internet sejak tahun 1990. Pada bagian ini akan dijelaskan bagaimana protokol HTTP bekerja, dimulai dari model hubungan HTTP, format pesan, dan cache di protokol HTTP.

RFC 1945 yang mendefinisikan protokol HTTP versi 1.0 ternyata dianggap masih memiliki kekurangan dan kemudian IETF menspesifikasikan protokol HTTP versi baru, yaitu 1.1 dalam RFC 2068. Saat buku ini ditulis RFC 2068 sedang berada dalam status *proposed standard* dan telah diimplementasikan dalam browser-browser versi 4. Perbaikan atas HTTP 1.0 antara lain adalah koneksi persistent dan pipelined serta model cache yang lebih baik.

8.3.1. Model hubungan HTTP

Protokol HTTP bersifat request-response, yaitu dalam protokol ini client menyampaikan pesan request ke server dan server kemudian memberikan response yang sesuai dengan request tersebut. Request dan response dalam protokol HTTP disebut sebagai request chain dan response chain. Hubungan HTTP yang paling sederhana terdiri atas hubungan langsung antara *user agent* dengan server asal. Hubungan HTTP tidak selalu seperti ini karena spesifikasi HTTP mengenal adanya beberapa komponen yang dapat terlibat dalam

membentuk sebuah hubungan HTTP, yaitu client, *user agent*, server asal, proxy, gateway, dan tunnel.

Menurut spesifikasi HTTP, istilah-istilah di atas adalah sebagai berikut:

Client
Program yang membentuk hubungan HTTP dengan tujuan untuk mengirimkan request

User agent
Client yang melakukan request, dapat berupa browser, editor, spider, atau perangkat lain

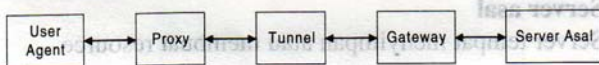
Server asal
Server tempat menyimpan atau membuat resource

Proxy
Program perantara yang bertindak sebagai server dan client dengan tujuan untuk membuat request atas nama client yang lain

Gateway
Server yang bertindak sebagai perantara untuk server lain. Gateway menerima request seolah-olah ia adalah server asal dan client tidak mengetahui bahwa gateway yang menerima request yang dikirim.

Tunnel
Program perantara yang bertindak sebagai perantara buta antara dua hubungan HTTP. Tunnel tidak dianggap sebagai pihak yang terlibat dalam hubungan HTTP, walaupun ia dapat membuat HTTP request.

Pada protokol HTTP terdapat tiga jenis hubungan dengan perantara: proxy, gateway, dan tunnel. Proxy bertindak sebagai agen penerus, menerima request dalam bentuk *Uniform Resource Identifier* (URI)

absolut, mengubah format request, dan mengirimkan request ke server yang ditunjukkan oleh URI. Gateway bertindak sebagai agen penerima dan menerjemahkan request ke protokol server yang dilayaninya. Tunnel bertindak sebagai titik relay antara dua hubungan HTTP tanpa mengubah request dan response HTTP. Tunnel digunakan jika komunikasi perlu melalui sebuah perantara dan perantara tersebut tidak mengetahui isi dari pesan dalam hubungan tersebut. Gambar contoh hubungan HTTP yang melibatkan beberapa komponen dapat dilihat pada Gambar 8.6 di bawah ini.

Gambar 8.6 Komponen dalam rantai request/response HTTP

Proxy atau gateway dapat menggunakan mekanisme cache untuk memperpendek rantai hubungan HTTP. Proxy pada gambar di atas dapat menggunakan cache dan memberikan response cache sehingga request dari UA tidak perlu dilayani oleh server asal karena proxy telah memberikan response atas request tersebut.

8.3.2. Hubungan Persistent

Perbedaan mendasar antara HTTP/1.1 dengan HTTP/1.0 adalah penggunaan hubungan persistent. Jika HTTP/1.0 membutuhkan sebuah koneksi TCP untuk setiap permintaan URI, maka HTTP/1.1 dapat menggunakan sebuah koneksi TCP saja untuk beberapa permintaan URI. Server HTTP/1.1 dapat

mengasumsikan bahwa hubungan yang digunakan dengan client HTTP/1.1 adalah hubungan persistent, kecuali jika client menyatakan tidak hendak menggunakan hubungan persistent. Dalam mekanisme ini, server dan client dapat mengirim sinyal untuk menutup koneksi TCP menggunakan header Connection: close. Setelah sinyal ini dikirim, client tidak boleh lagi mengirimkan request ke server.

Server atau client yang berhubungan menggunakan protokol HTTP dengan versi lebih rendah dari 1.1 harus tidak mengasumsikan terjadinya hubungan persistent kecuali dinyatakan dengan jelas melalui sinyal Connection: Keep-Alive.

Hubungan persistent juga mendukung request yang di-"*pipeline*", yaitu client mengirimkan beberapa request sekaligus tanpa menunggu response selesai datang dari server. Server yang menerima request yang di-"*pipeline*" harus memberikan response sesuai urutan request. Client yang mendukung pipeline harus siap untuk kembali mengirimkan request tersebut jika server tidak memberikan response.

8.3.3. Format HTTP

Kita mengenal protokol HTTP menggunakan format URL (Universal Resource Locator) HTTP dalam bentuk:

"http:" "//" host [":"port] [abs_path]
host adalah nama domain Internet yang legal.
port adalah bilangan yang menunjukkan port HTTP di host, jika port tidak disebutkan maka port HTTP diasumsikan sebagai 80.
abs_path menyatakan lokasi resource di dalam host.
Contoh:

http://khensu.cnrg.net:8000/~cleo/home.html
http://khensu.cnrg.net/bgp.html

Jika kita mengisikan URL tersebut ke browser, browser bertugas untuk mengartikan URL tersebut dan menerjemahkannya dalam komunikasi protokol HTTP. Aturan dalam mengartikan format URL HTTP mengikuti aturan umum URI, yaitu case-sensitive, kecuali nama dan skema URI case-insensitive. Dengan aturan ini maka http://khensu.CNRG.net:8000/~cleo/home.html sama dengan http://khensu.cnrg.net:8000/~cleo/home.html tetapi berbeda dengan http://khensu.cnrg.net:8000/~cleo/Home.html.

Komunikasi protokol HTTP terdiri atas pesan request yang diberikan oleh *user agent* dan response yang dikeluarkan oleh server. Setiap request dan response HTTP menggunakan format pesan generik seperti yang didefinisikan oleh RFC 822. Pesan HTTP terdiri atas baris mulai, header pesan, dan isi pesan beserta entity (opsional). Header pesan dan isi pesan dipisahkan oleh sebuah baris kosong, yaitu hanya berisi karakter CRLF. Baris mulai pada pesan request berisi pesan permintaan dari client, sementara pada pesan response, baris ini berisi status response atas request yang diterima. Header pesan dapat terdiri atas beberapa baris, bergantung pada field-field yang perlu disertakan dalam header tersebut. Terdapat empat jenis header pesan, yaitu header pesan umum yang berlaku di setiap jenis pesan, header request, header response, dan header entity.

Header yang umum pada pesan HTTP request dan response adalah sebagai berikut:

```
Header-umum  = Cache-Control | Connection | Date | Pragma
             | Transfer-Encoding | Upgrade | Via
```

Field Cache-control memberikan aturan yang harus ditaati oleh seluruh mekanisme cache dalam rantai request/response. Field Connection mengatur tipe hubungan HTTP, apakah akan menggunakan hubungan persistent atau tidak. Field Date memberikan informasi mengenai waktu asal pesan. Field Transfer-Encoding menentukan jenis transfer yang diberikan kepada isi pesan agar dapat sampai dengan aman ke client. Field Upgrade pada pesan HTTP digunakan untuk mengganti protokol yang hendak digunakan, field ini berguna sebagai mekanisme transisi dari protokol HTTP/1.1 ke protokol tipe lain. Field Via digunakan oleh proxy dan gateway untuk memberitahu jalur yang digunakan dalam sebuah rantai request/response.

Isi pesan HTTP digunakan untuk mengirimkan isi entity. Keberadaan isi pesan dalam pesan request ditandai dengan adanya header Content-Length. Dalam pesan response, keberadaan isi pesan ini tergantung atas kode-status yang diberikan. Dalam sebuah pesan HTTP, header Content-Length dan Transfer-Encoding tidak boleh muncul bersama-sama. Kedua header ini menunjukkan hal yang berlawanan: adanya header Transfer-Encoding menunjukkan bahwa panjang isi pesan tidak diketahui sementara header Content-Length menunjukkan panjang isi pesan dalam byte. Jika dalam sebuah pesan terdapat kedua header ini, maka header Content-Length harus diabaikan.

Setiap request dan response dalam komunikasi HTTP harus menyertakan versi protokol yang digunakan. Format penulisan versi protokol ini adalah

HTTP/<major>.<minor>

Major dan minor adalah bilangan yang menunjukkan versi dari protokol HTTP. Dengan aturan ini penulisan versi protokol untuk versi 1.0 adalah HTTP/1.0 dan HTTP/1.1 untuk versi 1.1.

Penyertaan versi protokol ini diperlukan karena dapat saja terjadi dalam sebuah hubungan HTTP, server dan client menggunakan versi yang berbeda. Jika ini terjadi maka server dan client harus menggunakan versi HTTP tertinggi yang mungkin agara keduanya dapat berkomunikasi dengan baik.

8.3.4. Request

Format baris mulai dari pesan request HTTP dimulai dengan metode request, diikuti oleh URI untuk request, versi protokol yang digunakan dan diakhiri oleh karakter CRLF

Baris-request = Method SP Request-URI SP HTTP-Version CRLF

Method menunjukkan metode apa yang hendak dilakukan atas resource yang ditunjuk oleh Request-URI. Ada beberapa metode yang didefinisikan oleh HTTP/1.1, yaitu: OPTIONS, GET, HEAD, POST, PUT, DELETE, dan TRACE. Untuk setiap pesan request, server harus memberikan kode jawaban untuk memberitahu apakah client diperbolehkan mengakses menggunakan method yang diinginkan. Jika method tidak boleh digunakan, server harus menjawab dengan

kode 405 (Method Not Allowed). Di antara metode-metode di atas, hanya metode GET dan HEAD yang harus diimplementasikan oleh semua server. Jika server tidak mengimplementasikan sebuah metode atau tidak mengenal metode yang diminta client, maka server harus memberikan response 501 (Not Implemented).

Metode GET mengambil informasi apa saja dalam bentuk entity yang diidentifikasikan oleh Request-URI. Metode HEAD sama dengan metode GET kecuali bahwa server harus tidak mengirimkan entity yang ditunjukkan oleh Request-URI. Metode POST digunakan untuk meminta server menempatkan entity yang dikirim bersama request sebagai subordinat dari Request-URI yang dituju. Metode ini biasa digunakan dalam mengirimkan form.

Metode PUT meminta server untuk menempatkan entity request sebagai Request-URI. Perbedaan antara POST dan PUT adalah Request-URI dalam metode POST bertindak sebagai proses penerima data atau sebagai gateway, sementara dalam metode PUT, Request-URI adalah entity yang terdapat dalam request.

Metode DELETE meminta server untuk menghapus URI yang dikirim client. Client tidak dapat menjamin server berhasil melaksanakan request yang diminta. Metode TRACE digunakan untuk meminta loop-back pesan. Metode TRACE ini dapat digunakan untuk diagnostik.

Request-URI menunjukkan resource yang hendak diakses melalui pesan request. Request-URI dapat berupa URI absolut, path absolut, atau tanda asteriks

('*'). Request-URI tergantung pada request itu sendiri. Tanda asteriks '*' berarti bahwa request tidak merujuk pada resource tertentu, melainkan pada server. Request-URI asteriks hanya boleh digunakan jika metode yang digunakan tidak harus merujuk pada sebuah resource. Contoh:

```
OPTIONS * HTTP/1.1
```

Request-URI absolut diperlukan jika request dilakukan ke server proxy. Proxy dapat memberikan jawaban berdasarkan cache atau meneruskan request tersebut. Proxy boleh meneruskan request tersebut langsung ke server asal atau ke server proxy yang lain. Jika proxy meneruskan request ke server proxy lain, maka Request-URI tidak boleh diubah. Contoh request absolut:

```
GET http://khnemu.cnrg.net:8080 HTTP/1.1
```

Request dalam bentuk path absolut digunakan untuk menentukan resource pada server atau gateway. Jika sebuah request menggunakan Request-URI path absolut maka request juga harus mengirimkan lokasi jaringan URI dalam sebuah field header host. Dalam kasus seperti request di atas, proxy dapat menghubungi server *khnemu.cnrg.net* pada port 8080 dan mengirimkan request

```
GET / HTTP/1.1
Host: khnemu.cnrg.net:8080
```

Perhatikan bahwa dalam Request-URI path absolut informasi host tujuan juga turut disertakan. Server HTTP/1.1 harus mengetahui bahwa sebuah request ditentukan dengan melihat Request-URI dan header host. Server HTTP/1.1 yang tidak membolehkan

resource berbeda untuk field Host yang berbeda dapat mengabaikan field header Host tersebut. Jika server tersebut membolehkan resource yang berbeda maka server tidak boleh mengabaikan field header Host dalam Request-URI path-absolut. Jika server tersebut menerima Request-URI absolut, maka field header Host harus diabaikan. Jika ternyata host yang dirujuk pada Request-URI absolut atau path-absolut bukan host yang valid di server, maka server harus memberikan respon kesalahan 400 (Bad Request).

Setelah baris-request, client mengirimkan request header yang berisi informasi mengenai request atau mengenai client itu sendiri ke server. Header-header tersebut adalah:

```
Header-request = Accept I Accept-Charset I Accept-
Encoding
          I Accept-Language I Authorization I From
          I Host I If-Modified-Since I If-Match
          I If-None-Match I If-Range I If-Unmodified-Since
          I Max-Forwards I Proxy-Authorization I Range
          I Referer I User-Agent
```

Keterangan mengenai beberapa field header dapat dilihat pada Tabel 8.3.

Tabel 8.3 Keterangan field header request

Field Header	Keterangan
Accept	Menentukan tipe media yang dapat diterima sebagai respon dari server Contoh: Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, */*
Accept-Charset	Mengindikasikan set karakter yang dapat diterima sebagai respon Contoh: Accept-Charset: iso-8859-1,*,utf-8

Field Header	Keterangan
Accept-Language	Memperkecil jenis bahasa yang lebih disukai untuk diterima oleh client Contoh: Accept-Language: da, en-gb;q=0.8, en;q=0.7
Host	Menentukan host dan port tempat resource hendak diambil. Client HTTP/1.1 harus menyertakan field ini dalam setiap request. Contoh: Host: hathor.cnrg.net:8001
If-Modified-Since	Digunakan bersama metode GET memberikan kondisi bahwa jika resource yang hendak diakses belum diubah sejak waktu yang ditentukan oleh field ini, maka resource tersebut tidak akan dikirim oleh server, melainkan server memberikan response 304 (not modified) tanpa isi pesan. Contoh: If-Modified-Since: Sat, 21 Mar 1998 19:43:31 GMT
User-Agent	Berisi informasi mengenai *user agent* yang melakukan request ke server. Contoh: User-Agent: Lynx/2.7.1 libwww-FM/2.14

Contoh request lengkap yang berasal dari browser lynx yang berbasis teks di mesin Unix adalah seperti di bawah ini:

```
GET / HTTP/1.0
Host: hathor.cnrg.net:8001
Accept: text/html, text/plain, text/sgml, text/x-sgml, application/x-
wais-source, application/html, */*;q=0.001
Accept-Encoding: gzip, compress
Accept-Language: en
Negotiate: trans
User-Agent: Lynx/2.7.1 libwww-FM/2.14
```

Contoh request yang berasal dari Netscape Navigator 4.0:

```
GET / HTTP/1.0
Connection: Keep-Alive
User-Agent: Mozilla/4.01 [en] (Win95; I)
Host: hathor.cnrg.net:8001
Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, */*
Accept-Language: en
Accept-Charset: iso-8859-1,*,utf-8
```

Contoh request yang berasal dari MS Internet Explorer 4.0:

```
GET / HTTP/1.1
Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg,
application/vnd.ms-excel, application/msword,
application/vnd.ms-powerpoint, */*
Accept-Language: en-us
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0 (compatible; MSIE 4.0; Windows NT)
Host: hathor.cnrg.net:8001
Connection: Keep-Alive
```

8.3.5. Response

Setelah menerima request, server harus memberikan response HTTP atas request tersebut, yang terdiri atas baris status, header-header, dan isi pesan. Baris status berisi kode-status yang berupa kode tiga digit dan frasa-alasan, yaitu penjelasan singkat atas kode-status tersebut.

Digit pertama kode-status menentukan kelas dari response. Protokol HTTP/1.1 mendefinisikan 5 nilai untuk digit pertama:

1xx: Informational – request diterima, dan proses berlanjut
2xx: Success – request diterima dan dimengerti
3xx: Redirection – request membutuhkan tindakan lebih lanjut
4xx: Client Error - request mengandung sintaks yang salah
5xx: Server Error – server gagal melakukan tindakan sesuai request.

Server HTTP dapat menghasilkan kode-status selain yang didefinisikan dalam RFC sepanjang digit pertama kode-status tersebut dimengerti oleh aplikasi HTTP.

Format baris-status adalah sebagai berikut:

```
Baris-Status = HTTP-Version SP Status-Code SP Reason-
Phrase CRLF
```

Setelah baris response, server HTTP mengirimkan header-header response ke client. Header ini memberikan informasi mengenai server serta mengenai akses lebih lanjut ke URI.

```
Header-response = Age | Location | Proxy-Authenticate
                | Public | Retry-After | Server | Vary
                | Warning | WWW-Authenticate
```

Keterangan mengenai header-response dapat dilihat pada Tabel 8.4.

Tabel 8.4 Keterangan field header response

Field Header	Keterangan
Age	Perkiraan lama waktu sejak response atas URI diambil dari server asal. Header ini harus digunakan oleh cache HTTP/1.1.
Location	Digunakan untuk mengarahkan client ke URL lain. Contoh: Location: http://www.w3.org/pub/WWW/People.html
Proxy-Authenticate	Field ini memungkinkan client untuk mengirimkan identitasnya untuk menggunakan proxy yang membutuhkan autentikasi.
Public	Memperlihatkan jenis metode yang didukung oleh server

Field Header	Keterangan
Retry-After	Dapat digunakan bersama kode-status 503 untuk menunjukkan berapa lama layanan belum dapat diberikan. Contoh: Retry-After: Fri, 31 Dec 1999 23:59:59 GMT
Server	Berisi informasi mengenai server yang menangani request Contoh: Server: Apache/1.2.4 FrontPage/3.0.3
Vary	Digunakan server untuk mengirimkan sinyal bahwa entity yang dikirim adalah hasil negosiasi yang 'server driven'. URI dengan header ini harus tidak di-cache.
Warning	Digunakan untuk memberi informasi tambahan selain dari kode-status.
WWW-Authenticate	Dikirimkan beserta kode-status 401. Terdiri atas setidaknya satu challenge yang menentukan scheme autentikasi dan parameter-parameternya.

Contoh pasangan request dan response HTTP adalah seperti di bawah ini

Request	Response
HEAD / HTTP/1.1 Host: khnemu.cnrg.net Connection: close	HTTP/1.1 200 OK Date: Thu, 19 Mar 1998 00:22:06 GMT Server: Apache/1.2.4 FrontPage/3.0.3 Last-Modified: Tue, 17 Mar 1998 07:56:48 GMT ETag: "4d1a4-157-350e2cc0" Content-Length: 343 Accept-Ranges: bytes Connection: close Content-Type: text/html

Request	Response
HEAD / HTTP/1.1	HTTP/1.1 400 Bad Request Date: Thu, 19 Mar 1998 00:23:59 GMT Server: Apache/1.2.4 FrontPage/3.0.3 Connection: close Content-Type: text/html

8.3.6. Entity

Setiap pesan HTTP baik request maupun response dapat menyertakan isi pesan atau entity tergantung dari apakah pesan tersebut memungkinkan untuk membawa entity. Entity HTTP terdiri atas header entity dan isi entity. Header entity berisi informasi mengenai isi entity atau mengenai resource yang ditunjuk oleh Request-URI.

```
Header-entity = Allow | Content-Base | Content-Encoding
              | Content-Language | Content-Length | Content-Location
              | Content-MD5 | Content-Range | Content-Type
              | ETag | Expires | Last-Modified
              | extension-header
```

Field Header	Keterangan
Allow	Menunjukkan metode yang didukung oleh resource yang ditunjuk oleh Request-URI
Content-Base	Menentukan base URI yang digunakan untuk mengartikan URI relatif. Contoh: Content-Base: http://amon.cnrg.net/
Content-Encoding	Menentukan tipe media dan mekanisme decoding yang harus diberikan pada entity. Contoh: Content-Encoding: gzip
Content-Language	Menentukan bahasa yang hendak digunakan.

Field Header	Keterangan
Content-Length	Menentukan panjang isi pesan dalam oktet
Content-Location	Menginformasikan lokasi resource dari isi pesan
Content-MD5	Untuk memeriksa integritas isi entity
Content-Range	Digunakan untuk mengirimkan sebagian dari seluruh entity. Misal untuk mengirim 500 byte pertama dari 1234 byte: Content-Range: bytes 0-499/1234
Content-Type	Menentukan tipe media isi entity yang dikirim ke penerima
ETag	Menentukan tag untuk entity yang bersangkutan. Digunakan untuk GET bersyarat.
Expires	Mengindikasikan waktu kadaluwarsa entity. Cache harus mengambil ke server asal jika waktu kadaluwarsa entity sudah terlewati.
Last-Modified	Mengindikasikan waktu terakhir perubahan entity.

Pesan HTTP dengan entity adalah seperti di bawah ini.

Contoh response:

```
HTTP/1.1 200 OK
Date: Thu, 19 Mar 1998 01:08:43 GMT
Server: Apache/1.2.4 FrontPage/3.0.3
Transfer-Encoding: chunked
Content-Type: text/html
1d8
<HTML><HEAD><TITLE>Please login to authentication server</TITLE>
<META HTTP-EQUIV="Expires" CONTENT="Thu, 19 Mar 1998 01:08:51 GMT">
</HEAD>
<BODY BGCOLOR="#000000" TEXT="#FFFFFF"><PRE>
```

```
<FORM METHOD="POST" ACTION="/cgi-bin/login">Username:
<INPUT NAME="login" SIZE=8 VALUE="">
<BR>Password: <INPUT TYPE="password" NAME="pass"
SIZE=24 VALUE="">
<BR><INPUT TYPE="hidden" NAME="new"
VALUE="1"> <BR>       <INPUT TYPE="submit"
VALUE="Login">
</FORM></PRE></BODY></HTML>
0
```

Contoh request:

```
POST /cgi-bin/logincgi HTTP/1.1
Referer: http://hathor.cnrg.net/~husni/postlogin.html
Connection: Keep-Alive
User-Agent: Mozilla/4.01 [en] (Win95; I)
Host: hathor.cnrg.net
Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, */*
Accept-Language: en
Accept-Charset: iso-8859-1,*,utf-8
Content-type: application/x-www-form-urlencoded
Content-length: 23
login=husni&pass=Ola
```

8.3.7. Cache HTTP

Salah satu cara untuk mempercepat response HTTP di jaringan adalah dengan menggunakan cache. Dalam protokol HTTP dimungkinkan dalam sebuah rantai request/response terdapat beberapa proxy yang dapat bertindak sebagai server cache. Spesifikasi protokol HTTP juga menyertakan elemen-elemen agar cache dapat dilakukan sebaik mungkin. Tujuan adanya cache adalah mencegah pengiriman request dan menghilangkan kebutuhan untuk mengirim response lengkap dari server. Cache mencegah pengiriman request ke server asal dengan menggunakan mekanisme kadaluwarsa sehingga memperpendek rantai request/response dan waktu round-trip. Menghilangkan kebutuhan untuk

mengirim response lengkap dari server asal dengan menggunakan mekanisme validasi berarti mengurangi kebutuhan bandwidth.

Isu utama dalam cache adalah kinerja, yang diukur dalam kecepatan waktu response, dan benarnya response yang dihasilkan cache. Cache harus memberi response yang paling 'up-to-date' atas sebuah request, yaitu response yang telah divalidasi dengan server asal; atau masih 'cukup segar'; atau memberikan peringatan jika 'kesegaran' cache dilanggar; atau response 304 (Not Modified), 305 (Proxy Redirect), atau error yang sesuai.

Mekanisme waktu kadaluwarsa dalam cache HTTP/1.1 dilakukan dengan menggunakan pendekatan heuristik. Cache menghitung umur entri URI dalam cache dan memperkirakan apakah entri tersebut sudah kadaluarsa. Cache HTTP/1.1 menggunakan header Age dalam berkomunikasi antar cache untuk menginformasikan selang waktu sejak sebuah entri diambil dari server asal. Dengan cara ini cache dapat menghitung umur sebuah entri yang diperolehnya dari cache lain.

Jika entri dalam cache dianggap tidak 'segar' lagi maka cache perlu melakukan validasi ke server asal atau server cache lain yang memiliki entri cache yang masih baru. Cache mem-validasi entri dengan mengirimkan request bersyarat ke server. Jika entri tersebut tidak berubah maka server akan mengeluarkan kode-status seperti 304 (Not Modified). Jika ternyata entri berubah, server akan mengirimkan isi pesan entri tersebut secara lengkap.

Cache menggunakan header ETag dan sebagai alat memvalidasi entri URI yang utama. ETag disebut

sebagai pemvalidasi kuat (*strong validator*) karena ETag berubah setiap kali isi pesan berubah. Parameter lain yang dapat digunakan sebagai alat pemvalidasi adalah waktu modifikasi terakhir. Parameter ini menunjukkan waktu terakhir entity tersebut berubah sampai hitungan detik. Parameter ini disebut sebagai pemvalidasi lemah (*weak validator*) karena nilainya tidak selalu berubah walaupun entity berubah (entity dapat saja berubah dua kali dalam satu detik). Dalam beberapa kasus server tidak ingin menggunakan ETag sebagai pemvalidasi kuat tersebut karena entity tidak berubah secara signifikan.

Mekanisme validasi dan waktu kadaluarsa juga dapat diatur oleh server asal atau client melalui header Cache-Control. Header Cache-Control yang terdapat dalam pesan HTTP akan menyebabkan cache mengikuti mekanisme validasi dan kadaluwarsa yang sesuai dengan header tersebut. Client dapat memaksa cache untuk memvalidasi entri ke server asal dengan header

```
Cache-Control: max-age=0
```

atau memaksa agar mengambil entri terbaru dari server asal dengan header

```
Cache-Control: no-cache
```

Server juga dapat memaksa agar cache HTTP/1.1 selalu memvalidasi sebuah request ke server asal dengan header

```
Cache-Control: must-revalidate
```

yang dikirim beserta response atas request tersebut.

Untuk melihat bagaimana cache HTTP bekerja, di bawah ini diperlihatkan contoh request dari client melalui server proxy. Pada request pertama, client meminta URL http://khensu.cnrg.net/ melalui server cache *proxy.cache.net*. Isi request dari client adalah seperti di bawah ini.

```
GET http://khensu.cnrg.net/ HTTP/1.1
Host: khensu.cnrg.net
Connection: close
```

Header dari response server cache adalah seperti berikut:

```
HTTP/1.0 200 OK
Date: Fri, 20 Mar 1998 10:08:43 GMT
Server: Apache/1.2.4 FrontPage/3.0.3
Last-Modified: Tue, 17 Mar 1998 07:56:48 GMT
ETag: "4d1a4-157-350e2cc0"
Content-Length: 343
Accept-Ranges: bytes
Content-Type: text/html
X-Cache: MISS from proxy.cache.net
<ISI PESAN RESPONSE>
```

Dari response yang diberikan oleh server *proxy.cache.net* terlihat bahwa server tersebut tidak menyimpan URL yang diminta client. Hal ini terlihat melalui header X-Cache: MISS.

Beberapa saat kemudian client kembali melakukan request atas URL yang sama. Server proxy memberikan response dan kali ini server tersebut menjawab dengan menyatakan bahwa URL tersebut tersimpan dalam cache.

```
HTTP/1.0 200 OK
Date: Fri, 20 Mar 1998 10:08:43 GMT
Server: Apache/1.2.4 FrontPage/3.0.3
```

```
Last-Modified: Tue, 17 Mar 1998 07:56:48 GMT
ETag: "4d1a4-157-350e2cc0"
Content-Length: 343
Accept-Ranges: bytes
Content-Type: text/html
X-Cache: HIT from proxy.cache.net
<ISI PESAN RESPONSE>
```

Setelah selang beberapa waktu client kembali meminta URL yang sama, dan kali ini client memaksa agar server cache melakukan validasi ke server asal dengan menggunakan header Cache-Control. Request dari client adalah seperti berikut:

```
GET http://khensu.cnrg.net/ HTTP/1.1
Host: khensu.cnrg.net
Cache-Control: max-age=0
Connection: close
```

Karena client memaksa server cache untuk memvalidasi URL ke server asal, maka server cache melakukan request bersyarat ke server asal tersebut. Alat validasi yang digunakan oleh server cache adalah tanggal perubahan terakhir URL, seperti request berikut:

```
GET / HTTP/1.0
If-Modified-Since: Tue, 17 Mar 1998 07:56:48 GMT
Host: khensu.cnrg.net
Cache-Control: max-age=0
Via: 1.1 proxy.cache.net:3128 (Squid/1.2.beta18)
X-Forwarded-For:
Connection: Keep-Alive
```

Setelah mendapat response dari server asal, server cache memberikan response ke client seperti berikut:

```
HTTP/1.0 200 OK
Date: Fri, 20 Mar 1998 11:08:02 GMT
Server: Apache/1.2.4 FrontPage/3.0.3
```

```
Last-Modified: Tue, 17 Mar 1998 07:56:48 GMT
ETag: "4d1a4-157-350e2cc0"
Content-Length: 343
Accept-Ranges: bytes
Content-Type: text/html
X-Cache: MISS from proxy.cache.net
<ISI PESAN RESPONSE>
```

Response yang dikeluarkan server cache menunjukkan bahwa URL tersebut tidak berasal dari cache yang disimpan server, melainkan langsung dari server asal walaupun tanggal URL terakhir diubah serta ETag tetap sama dengan response sebelumnya.

8.4. Ringkasan

Aplikasi merupakan protokol level atas dalam lapisan protokol TCP/IP. Beberapa aplikasi penting antara lain SNMP, FTP, SMTP, dan HTTP. SNMP telah dibahas di bab sebelumnya. FTP merupakan protokol aplikasi untuk keperluan transfer data dari komputer satu ke komputer lain. SMTP digunakan untuk mengirim pesan dan juga file antar pengguna Internet yang memiliki alamat e-mail. Sedangkan HTTP merupakan protokol untuk mentransfer hypertext yang banyak digunakan oleh pengguna saat melakukan browsing.

Aplikasi-aplikasi di atas bersifat client-server. Artinya, dalam setiap operasinya, selalu ada mesin yang bertindak sebagai server, yang tugasnya sebagai penyedia data (FTP, HTTP) atau layanan (SMTP). Komunikasi antara client dan server inilah yang diatur oleh protokol aplikasi tersebut. Ketiga protokol yang dijelaskan di atas menggunakan sistem *request-response*, cara yang paling umum digunakan dalam hubungan client-server.

Setiap kali client mengirimkan request, server selalu terlebih dahulu memberikan baris status agar client dapat menentukan apa yang selanjutnya dilakukan terhadap response tersebut.

Berbeda dengan protokol SMTP dan HTTP, protokol FTP membuka dua buah hubungan TCP karena memerlukan hubungan kontrol dan hubungan data yang hendak memaksimalkan kecepatan pengiriman file. Perkembangan berikutnya dari SMTP dan HTTP adalah penggunaan *connection caching*, yaitu mengirimkan beberapa set data (mail untuk SMTP, URI untuk HTTP) sekaligus dalam satu kali hubungan TCP. Dengan menggunakan connection caching terbukti kecepatan pengiriman data lebih cepat dibandingkan tanpa conection caching.